PLM-järjestelmän muutosten automaattinen dokumentointi

Menetelmiä muutosdatan käsittelyyn liiketoimintajärjestelmässä

Ammattikorkeakoulun opinnäytetyö
Tieto- ja viestintätekniikka, insinööri (AMK)
Kevät 2025
Antti Vuorio

HAMK
Hämeen ammatti-
korkeakoulu

Tiivistelmä

Koulutus	Tieto- ja viestintätekniikka, insinööri (AMK)	
Tekijä	Antti Vuorio	Vuosi 2025
Työn nimi	PLM-järjestelmän muutosten automaattinen dokumentointi	
Ohjaaja	Petri Kuittinen (HAMK), Michael Nyman (Symetri Oy)	

Opinnäytetyö pureutuu keskeiseen haasteeseen tuotteen elinkaaren hallintajärjestelmien (PLM) dokumentoinnissa: miten seurata ja tallentaa asiakaskohtaisten järjestelmäasennusten muutoksia tehokkaasti ja järjestelmällisesti. Tutkimus kohdistuu Sovelia Core PLM -järjestelmään, jonka muutosten havainnointiin ei ole aiemmin ollut kattavaa automaattista seurantamenetelmää. Opinnäytetyö toteutettiin toimeksiantona Symetri Oy:lle, jonka tuote Sovelia Core PLM on.

Tutkimuksen lähtökohtana oli tunnistaa nykyisten dokumentointimenetelmien puutteet. Vaikka versionhallinta ja manuaaliset dokumentit ovat olleet käytössä, ne eivät tarjoa riittävän kattavaa ja selkeää kuvaa järjestelmän muutoksista. Ratkaisussa käytettävien menetelmien tutkimuksessa etsittiin vastauksia kolmeen aiheeseen: kerättävien parametrien valintaan, sekä tiedon keräämisen ja tallentamisen menetelmiin.

Toteutusvaiheessa keskityttiin erityisesti järjestelmän pienten automaatioiden, kytkinten, käytön seurantaan. Tutkimus osoitti, että nykyisellä järjestelmäkonfiguraatiolla voidaan kerätä merkittävää tietoa asiakaskohtaisten asennusten kehittämiseksi. Kehitetty ratkaisu muodostuu kahdesta komponentista: komentorivityökalu lokianalysaattorista ja web-pohjaisesta visualisointityökalusta.

Automaattisen seurantajärjestelmän hyödyt ovat moninaiset. Ne ulottuvat tarkemmasta muutosten dokumentoinnista ajansäästöön rutiinitehtävissä ja mahdollistavat paremman historiatietojen saatavuuden. Erityisen arvokas on kyky ennakoida järjestelmän tulevia tarpeita ja tunnistaa kehitystrendejä.

Tutkimus tunnisti useita kehittämiskohteita, kuten lokitiedostojen tietosisällön optimointi, tiedonhallinnan skaalautuvuuden parantaminen ja automaatiotason nostaminen. Jatkotutkimukselle avautuu lupaavia polkuja pilvipohjaisiin datajärjestelmiin, edistyneisiin dataputkiarkkitehtuureihin ja koneoppimisen hyödyntämiseen lokien analysoinnissa.

Johtopäätöksenä tutkimus tarjoaa lupaavan lähtökohdan automaattiselle dokumentointijärjestelmälle, joka voi merkittävästi parantaa PLM-järjestelmien hallintaa ja asiakaskohtaista palvelua.

Avainsanat	PLM, Tuotteen elinkaaren hallinta, dokumentaatio, datan käsittely
Sivut	30 sivua ja liitteitä 5 sivua

HAMK
Hämeen ammatti-
korkeakoulu

Abstract

DP Information and Communication Technology
Author Antti Vuorio
Year 2025
Subject PLM System Automated Change Documentation
Supervisors Petri Kuittinen (HAMK), Michael Nyman (Symetri Oy)

The thesis examines a significant challenge in the documentation of Product Lifecycle Management (PLM) systems: how to efficiently and systematically track and record changes in customer-specific system installations. The research focuses on the Sovelia Core PLM system, which previously lacked a comprehensive automatic tracking method for changes. The Sovelia Core PLM system is a product of Symetri Oy, the commissioner of this thesis.

The starting point of the research was to identify the shortcomings of current documentation methods. Although version control and manual documents have been used, they do not provide a sufficiently comprehensive and clear picture of changes in the system. The research into methods for the solution sought answers to three topics: the selection of parameters to be collected, and the methods for collecting and storing information.

In the implementation phase, special attention was given to monitoring the use of small automations called triggers. The research demonstrated that with the current system configuration, it is possible to collect significant information to improve customer-specific installations. The developed solution consists of two main components: a command line log analyser and a web-based visualization tool.

The benefits of the automatic tracking system are manifold. They range from more precise change documentation to saving time in routine tasks and enabling better access to historical data. Particularly valuable is the ability to anticipate the system’s future needs and identify development trends.

The research identified several areas for improvement, such as optimizing the data content of log files, enhancing the scalability of data management, and increasing the level of automation. Promising avenues for further research include cloud-based data systems, advanced data pipeline architectures, and leveraging machine learning in log analysis.

As a conclusion, the research provides a promising starting point for an automatic documentation system that can significantly enhance the management of PLM systems and customer-specific service.

Keywords PLM, Product Lifecycle Management, documentation, data processing
Pages 30 pages and appendices 5 pages

Sisällys

Kuvat

Taulukot

Liitteet

Lyhenteet

BI	(Business Intelligence) Liiketoimintatiedon hallinta
CAD	(Computer-aided design) Tietokoneavusteinen suunnittelu
CRM	(Customer relationship management) Asiakkuuden hallinta
ECO	(Engineering Change Order) Teknisen muutoksen tilaus
ECR	(Engineering Change Request) Tekninen muutospyyntö
ERP	(Enterprise resource planning) Toiminnanohjaus
ETL	(Extract, transform, load) Datan purkaminen, muuntaminen, lataaminen
IoT	(Internet of Things) Esineiden internet
PDM	(Product Data Management) Tuotetiedon hallinta
PLM	(Product Lifecycle Management) Tuotteen elinkaaren hallinta

Termit

Big Data	Suuri datamäärä, jota ei voi hallita perinteisillä datanhallinnointitavoilla
Data Engineering	Järjestelmien kehittäminen tietojen keräämisen ja käytön mahdollistamiseksi
Datajärvi	Keskitetty tietovarasto, joka mahdollistaa erityyppisten tietojen keräämisen ja tallentamisen jatkokäsittelyä varten
Dataputki	Useasta lähteestä kerätty raakadata siirretään datavarastoon, kuten datajärveen tai datavarastoon
Datavarasto	Suuri tietovarasto, joka on kerätty useista lähteistä

Nimike Tuotetiedon perusyksikkö

PostgreSQL Avoimen lähdekoodin tietokannan hallintajärjestelmä

Tiedonlouhinta Menetelmät oleellisen tiedon löytämiseen suuresta datamäärästä

Tuote Valmistettava hyödyke, joka voi olla kokonainen kone tai sen osa

Tuotetiedon hallinta Prosessi, jolla keskitetään yrityksen tuottamien tuotteiden tieto

Tuotteen elinkaaren hallinta Prosessi, joka pitää sisällään tuotetiedon hallinnan sekä huomioi lisäksi tuotteen elinkaaren

1 Johdanto

Tämän opinnäytetyön päätavoite on määritellä tehokas ja automaattinen järjestelmä Sovelia Core PLM -asennuksissa tapahtuvien muutosten seurantaan, joka tallentaa määriteltyjen parametrien arvot säännöllisin väliajoin helposti hyödynnettävässä muodossa. Tutkimus selvittää dokumentoitavien tietojen sisällön, keräysmenetelmät sekä tehokkaat varastointi- ja esitystavat, jotta palveluorganisaatio voi hyödyntää tietoja asiakaskohtaisten järjestelmäasennusten kehitystyössä.

Opinnäytetyön keskeisen tutkimuskysymyksen voi muotoilla tiiviisti: "Miten Sovelia Core PLM:stä kannattaisi kerätä muutoksiin liittyvää dataa, jotta se auttaisi tehostamaan asiakaskohtaisen järjestelmäasennuksen kehitystä asiakkaan uudet tarpeet täyttäväksi?"

Tuotteen elinkaaren hallinta (Product Lifecycle Management, PLM) kattaa tuotteen koko elinkaaren suunnittelusta ja valmistuksesta aina varaosatoimituksiin saakka. PLM-järjestelmät ovat ohjelmistoratkaisuja, jotka tarjoavat keskitetyn alustan tuotetiedon hallintaan, erityisesti teollisuusyrityksille. Sovelia Core PLM on yksi tällainen järjestelmä, ja se tarjoaa valmiita työkaluja ja prosesseja, joita voidaan mukauttaa yrityksen tarpeisiin. Tämä tekee siitä joustavan ratkaisun monenlaisiin liiketoimintaympäristöihin. Tässä opinnäytetyössä tutkimuskohteena on Sovelia Core PLM-järjestelmäasennuksessa tapahtuvien muutosten seuranta. Opinnäytetyössä keskitytään erityisesti Sovelia Core PLM-järjestelmään, sillä opinnäytetyön tekijällä on käytännönläheistä työkokemusta kyseisen järjestelmän parissa. Opinnäytetyön tekemisen pohjana toimii tekijän tietotekniikan koulutus sekä työkokemus PLM-järjestelmän parissa. Näitä kokemuksia hyödynnetään myös opinnäytetyössä käytettävien lähteiden luotettavuuden ja oikeellisuuden arvioinnissa.

Asiakaskohtaisessa PLM-järjestelmäasennuksessa tapahtuvat muutokset ovat tärkeää tietoa sen kehitystarpeiden arvioinnin tueksi. Järjestelmään tehdään räätälöityjä muutoksia asiakastarpeitten mukaan. Muutostyöstä jää jälkiä versionhallintaan, mutta ne eivät anna riittävän tarkkaa kokonaiskuvaa järjestelmän tilasta muutosten jälkeen. Muutoksista järjestelmässä tarvitaan tietoa, jotta tuleviin asiakkaan tarpeisiin pystytään vastaamaan. Kehitystarpeiden tunnistaminen, ja sen myötä ratkaiseminen on tarpeettoman työlästä, sillä niiden tunnistamiseksi tarvittavaa hyvin jäsenneltyä tietoa on saatavilla rajallisesti. Osa tarvittavasta tiedosta on löydettävissä järjestelmästä, mutta se on tehtävä käsin. Näistä syistä tarvitaan erillinen tekninen ratkaisu, jolla seurataan järjestelmässä tapahtuvien olennaisimpien parametrien muutoksia, jotta saadaan muodostettua tilannekuva tehokkaasti.

Palveluorganisaatio tarvitsee tietoa muutoksista, jotta se kykenee tarjoamaan asiakkailleen mahdollisimman laadukasta palvelua. Opinnäytetyössä etsitään ratkaisuvaihtoehtoja tutkimuksen

kohteena olevien muutosten seurannan tehostamiseen. Tuloksena saatavat tiedot auttavat kehittämään järjestelmään toiminnallisuuden, joka mahdollistaa asiakaspalvelun tason nostamisen. Tässä opinnäytetyössä selvitetään sekä tarpeita kerättävän tiedon laadusta että siitä, miten sitä kerättäisiin, ja mihin sitä tallennettaisiin.

Opinnäytetyön tavoitteena on määritellä, mitä tietoja järjestelmästä pitäisi dokumentoida automaattisesti, ja millä tavoin se tehtäisiin. Tämän tavoitteen saavuttamiseksi tutkimus keskittyy keskeisiin kysymyksiin:

1. Mitä käyttötarkoituksia kerätylle datalle voidaan löytää palveluorganisaation näkökulmasta?
2. Miten vastaavaa dataa kerätään ja varastoidaan muissa järjestelmissä?
3. Mitkä ovat vaihtoehtoiskustannukset, jos dataa jätetään keräämättä automaattisesti Sovelia Core PLM:stä?
4. Miten dataa voidaan kerätä, varastoida ja visualisoida tehokkaasti Sovelia Core PLM:ssä?

Järjestelmän muutosten vaihtoehtoisia dokumentointimenetelmiä etsitään tutkimalla aiheesta toisaalla tehtyjä tutkimuksia. Lisäksi selvitetään, miten tutkimusaiheen kanssa sovelluskelpoisia ongelmia on ratkottu käytännössä muualla. Opinnäytetyö koskee Sovelia Core PLM-järjestelmän kehittämistä, siinä ei oteta kantaa järjestelmän käyttökokemukseen. Opinnäytetyö keskittyy Sovelia Core PLM-järjestelmää tarjoavan yrityksen oman toiminnan kehittämiseksi tarvittavaan tutkimukseen.

Opinnäytetyössä perehdytään aluksi tuotteen elinkaaren hallintajärjestelmän teoriaan sekä sen merkitykseen tuottavassa teollisuudessa. Sen jälkeen etsitään taustaksi toisaalla tehtyjä tutkimuksia sekä ratkaisuja opinnäytetyön aihepiiriin sovellettaviin ongelmiin. Viimeisenä tutkimuksen osana määritellään Sovelia Core PLM-järjestelmän avuksi kehitettävä automaattinen dokumentointijärjestelmä.

Tämän opinnäytetyön keskeisimmät kontribuutiot ovat sekä käytännöllisiä että menetelmällisiä. Ensinnäkin työ osoittaa konkreettisesti automaattisen dokumentoinnin hyödyllisyyden Sovelia Core PLM -ympäristössä luomalla prototyypin, joka todentaa lähestymistavan toimivuuden ja resurssien säästöpotentiaalin. Toiseksi opinnäytetyö tarjoaa tuotantokäyttöön soveltuvan arkkitehtuuriehdotuksen Azure-pohjaisine toteutusmahdollisuuksineen. Kolmanneksi työ yhdistää tiedonlouhinnan menetelmiä PLM-järjestelmäkontekstiin, mikä edustaa uudenlaista lähestymistapaa. Lisäksi työ tunnistaa selkeitä jatkokehityskohteita ja avaa uusia suuntia tutkimuksen jatkamiseen, kuten datajärvet ja koneoppimismenetelmät PLM-datan analysoinnissa. Erityisen arvokasta on liiketoiminnallisen lisäarvon osoittaminen sekä järjestelmätoimittajalle että loppuasiakkaille, luoden perustan datalähtöiselle PLM-järjestelmien kehittämiselle.

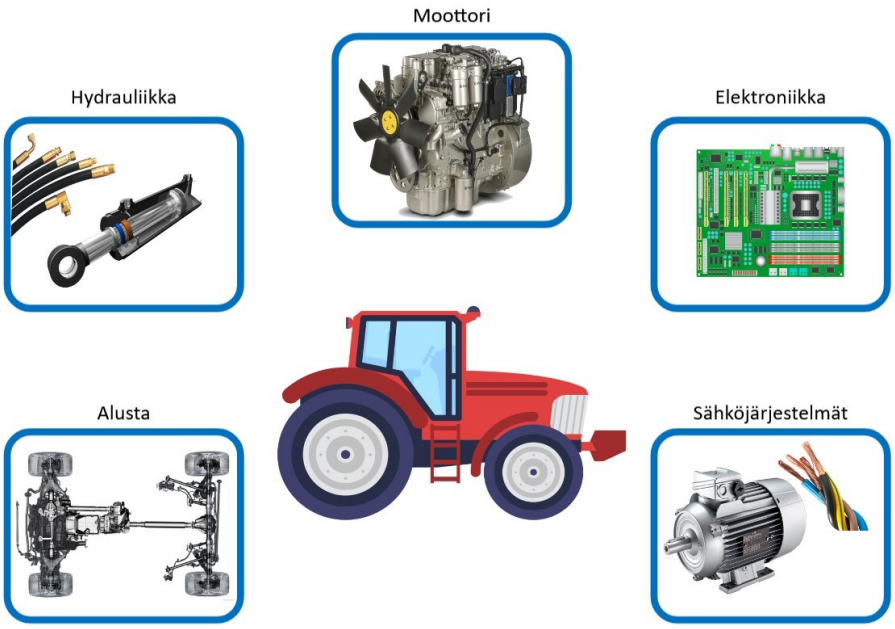
2 Tietoperusta

2.1 Alustus

Valmistus on prosessi, jossa raaka-aineet tai osat muutetaan valmiiksi tuotteeksi käyttämällä työkaluja, ihmistyötä, koneita ja kemiallista käsittelyä. Nykyisin valmistavassa teollisuudessa käytetään massatuotantoa, kokoonpanolinjavalmistusta sekä mekanisointia suurempien tavaramäärien valmistukseen halvemmalla (Kenton, 2024). Esimerkki valmistavan teollisuuden tuotteesta on traktori, joka on monimutkainen laite, joka koostuu useista järjestelmistä (kuva 1).

Traktorin, kuten muidenkin tuottavan teollisuuden tuotteiden valmistamiseen käytetään useita tietojärjestelmiä, joista yksi keskeinen on Tuotteen elinkaaren hallinta (PLM). Tässä opinnäytetyössä perehdytään PLM-järjestelmään osana tuottavan teollisuuden yrityksen prosesseja, sekä tutkitaan ratkaisumenetelmiä asiakasyrityskohtaisen PLM-järjestelmäasennuksen kehittämisen tehostamiseksi.

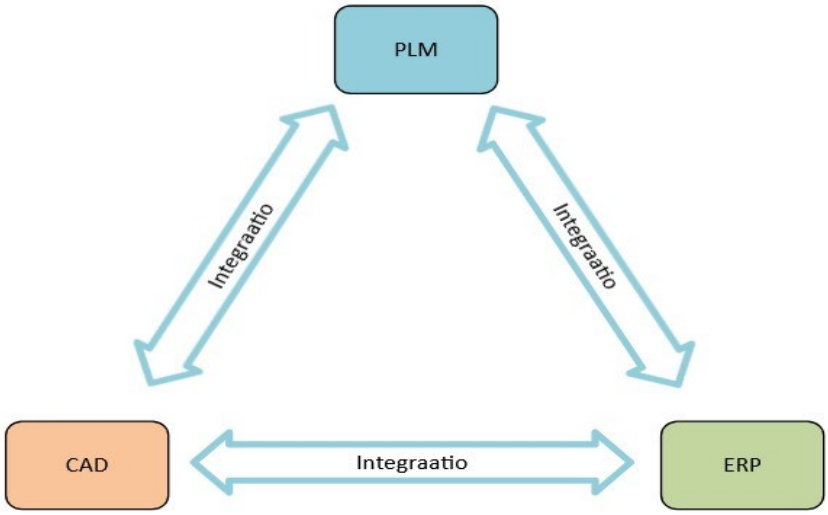
Kuva 1. Traktorin muodostavat tärkeimmät järjestelmät



2.2 Liiketoimintajärjestelmät

Tuottavan teollisuuden yritykset ohjaavat operatiivista toimintaansa tietojärjestelmillä, jotka ovat erikoistuneet tiettyihin liiketoiminnan kannalta tärkeisiin operaatioihin. Näiden yritysten käyttämistä

Kuva 2. Integraatiot liiketoimintajärjestelmien välillä



liiketoimintajärjestelmistä tuotannonohjausjärjestelmä (ERP), tuotteen elinkaaren hallintajärjestelmä (PLM), tietokoneavustettu suunnittelu (CAD) sekä asiakkuudenhallintajärjestelmä (CRM) ovat niille tärkeimpiä. PLM toimii keskitettynä arkistona ja siltana CAD:n ja ERP:in välillä. (3R, n.d.). Usein tuotavan teollisuuden yrityksellä voi olla kahdensuuntaiset integraatiot, eli tieto liikkuu molempiin suuntiin PLM:n, ERP:in sekä CAD:in välillä (kuva 2). Integraatiot voivat olla myös vaillinaisia, esimerkiksi PLM:stä ERP:iin voi olla vain yhdensuuntainen integraatio PLM → ERP.

CAD:issa tuotetaan tuotetietoa, jota myöhemmin käytetään PLM:stä käsin. ERP:iä pääasiallisena tietolähteenään käyttäviä käyttäjäryhmiä ovat esimerkiksi myyjät, jotka käyttävät ERP:istä saatavaa tietoa valmistettujen tuotteiden myyntiin. Ostajan roolissa oleva henkilö voi käyttää ERP:iä ostotilausten tekemiseen, mutta voi lisäksi tarvita ostoon liittyviä tietoja myös PLM:stä. Tiedot PLM:stä ostaja saattaa hakea PLM- ja ERP-konfiguraation mukaan joko PLM:n käyttöliittymästä suoraan, tai tarvittavat tiedot voidaan tuoda hänen saatavilleen PLM:stä integraation avulla ERP:iin. Koneen suunnittelija on tyypillinen käyttäjäryhmä, jolle PLM on suunnittelutyössä käytettävän CAD-ohjelmiston ohella tärkein järjestelmä. PLM toimii suunnittelijalle hänen työssään sekä tietolähteenä, että tallennuspaikkana hänen työssään tuottamalle tiedolle. PLM:ssa ja ERP:ssa suoritettavat toiminnot eivät ole kaikilta osin tarkasti sidotut tiettyyn järjestelmään, vaan organisaation mukaan tietyt toiminnot voidaan suorittaa joko PLM:ssa tai ERP:ssa. Organisaatiosta ja sen prosesseista riippuu, miten eri toimintojen suorittaminen eri järjestelmissä on tarkoituksenmukaisinta järjestää. ERP kuuluu tuottavan teollisuuden yrityksen

ensimmäisiin käyttöön otettaviin tietojärjestelmiin, PLM:n käyttöönotto voi tapahtua paljon myöhemmin. Tästä syystä PLM:ssä käyttöönotettavat toiminnallisuudet voivat riippua siitä, kuinka paljon toimintoja on ehditty ottaa käyttöön ERP:issä, tai jossain muussa järjestelmässä ennen PLM-järjestelmän käyttöönottoa.

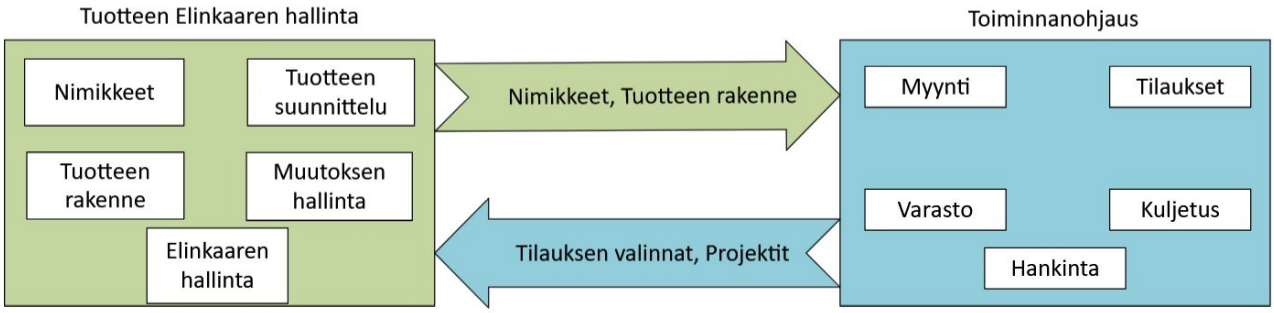
Yrityksen historia vaikuttaa järjestelmien käyttöönottoon merkittävästi. Jos yritys on aloittanut toimintansa esimerkiksi 1970-luvulla, se on voinut toimittaa ensimmäiset tuotteensa ilman minkäänlaisia tietojärjestelmiä piirtäen koneet käsin paperille. Ajan saatossa vähitellen käyttöön otetuilla järjestelmillä on myöhemmin alettu suorittaa niitä toimintoja, joita sen ajan järjestelmät ovat mahdollistaneet. 2020-luvulla perustettava uusi tuottavan teollisuuden yritys sen sijaan todennäköisesti ottaisi käyttöön kaikki neljä järjestelmää ERP, PLM, CAD ja CRM heti liiketoimintansa perustamisvaiheessa. Tuottavan teollisuuden yritysten välillä on iän lisäksi muitakin eroja, kuten tuoteportfolion laatu sekä yrityksen koko. Osa yrityksistä haluavat tietojärjestelmiensä ylläpidon paikallisena omilla palvelimilla, sen sijaan toiset yritykset erityisesti haluavat pilvipalveluna toimitettavan järjestelmän. Kaikki nämä tekijät vaikuttavat oleellisesti siihen, millaisia ominaisuuksia ne tarvitsevat tietojärjestelmiltä. (Morrison, 2025).

Erilaiset tarpeet tietojärjestelmiltä ovat sovellettavissa ERP:in lisäksi myös kaikkiin kolmeen muuhun tässä mainittuun tietojärjestelmään, myös PLM:ään. Näitten takia PLM-järjestelmiä kehittävät yritykset eivät voi realistisesti tehdä kaiken kattavaa järjestelmää, joka kykenee sellaisenaan vastaamaan kaikkiin niihin tarpeisiin, joita kaikki erilaiset PLM-järjestelmää hankkivat yritykset siltä haluavat. PLM-järjestelmää käyttöönotettaessa sen hankkivalta yritykseltä sekä sitä tarjoavilta yrityksiltä vaaditaan paljon tiivistä yhteistyötä, jotta hankkiva yritys saa lopulta käyttöönsä sen tarpeet täyttävän järjestelmän. PLM-järjestelmää – kuten muitakaan tässä luvussa mainittuja järjestelmiä ei voi yleensä ostaa ”kaupan hyllyltä” sellaisenaan ja ottaa heti käyttöön. Niiden käyttöönotossa täytyy huomioida kaikki ne tarpeet, joita täyttämään järjestelmät hankitaan.

Ensimmäinen käyttöönottoon vaikuttava tekijä on suunnittelijoiden käyttämä CAD-suunnitteluohjelmisto. PLM:ssä hallitaan CAD:ssa luotavaa tuotetietoa, joten näiden järjestelmien keskinäinen yhteensopivuus tulee varmistaa. Jos ne eivät jo valmiiksi ole yhteensopivia, ne täytyy sovittaa yhteen. Toiseksi täytyy varmistaa, että PLM:n ja ERP:n välillä liikkuva tieto (Kuva 3) liikkuu integraatiossa sellaisessa muodossa, että se tulkitaan molemmissa järjestelmissä oikein. Kolmanneksi kaikki työntekijät, joiden työtä PLM koskettaa täytyy kouluttaa käyttämään PLM:ää oikein. PLM-käyttöönottohankkeessa Sovelia Core PLM tapauksessa ennen varsinaista käyttöönottoa hankkijayritys ja toimittaja tekevät yhdessä vaatimusmäärittelyn. Siinä määritellään ne toiminnot, jotka järjestelmällä tulee voida suorittaa käyttöönoton valmistuttua. Sen jälkeen varsinaisessa käyttöönottoprojektissa lähtökohdaksi otetaan

Sovelia Core PLM-perusasennus, jonka pohjalta siitä räätälöidään sellainen kokonaisuus, joka täyttää aiemmin määritellyt vaatimukset.

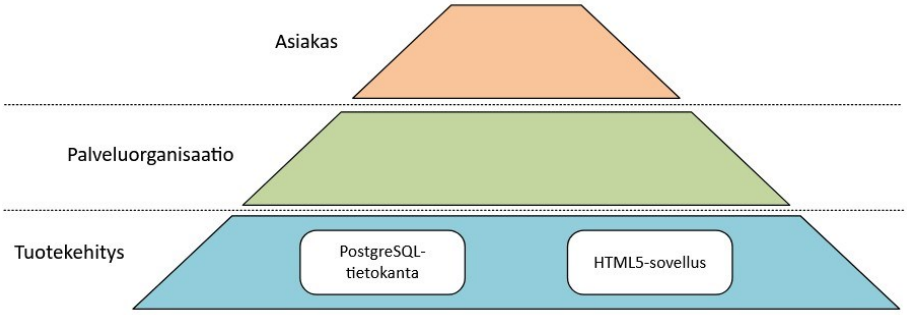
Kuva 3. Tiedon kulku PLM:n ja ERP:in välillä (Roima, n.d.)



Kuvassa 4 alin taso kuvaa Sovelia Core PLM-perusasennusta, jonka päälle rakennetaan asiakaskohtaisia toiminnallisuuksia. On melko tavallista, että käyttöönottoprojektin aikana hankkijan asettamat vaatimukset järjestelmälle muuttuvat – käytännössä usein vaatimusten määrä järjestelmälle kasvaa. Jotta järjestelmän käyttöönotto tuotannossa ei veny hallitsemattomasti vaatimuslistauksen paisuessa, sekä tilaajan että toimittajan täytyy yhdessä löytää kultainen keskitie: mitä toiminnallisuuksia täytyy ottaa ensimmäisenä käyttöön, mitkä voivat odottaa myöhempää käyttöönottoa.

Tällä tavoin uutta järjestelmää päästään käyttämään ja oppiminen sen käytöstä ja ominaisuuksista voi alkaa nopeammin. Muutosten pääkategorioiden muita tasoja käsitellään tarkemmin kappaleessa 2.6.

Kuva 4. Muutosten alin taso



2.3 Yleiset PLM-käyttötapaukset

Tässä kappaleessa esitellään käyttötapauksia, joissa tuottavan teollisuuden yritys voi hyödyntää PLM-järjestelmää tehostaakseen tuotantoprosessejaan. Tekninen muutostarve tuotteelle voi syntyä missä tahansa tuotteen elinkaaren vaiheessa. Vaikka uudelle julkaistavalle tuotesarjalle tehtäisiin kattavat

testaukset ennen sen asiakkaille toimittamisen aloittamista, siihen saattaa silti jäädä suunnitteluvirheitä. Virhe saatetaan huomata asiakkaan toimesta, kun tuote on jo otettu käyttöön sen varsinaisessa käyttötarkoituksessa. Silloin asiakas saattaa soittaa tuohtuneen puhelun tuotteen valmistaneelle yritykselle ja toivoa pikaista korjausta tilanteeseen. Tällaisessa tilanteessa tuotteen valmistanut yritys aloittaa teknisen muutospyyntöprosessin (ECR)(Rautelin, 2025). Sen tehtävä on välittää kaikki se informaatio, joka tarvitaan päätöksen tekoon muutoksen toteuttamiseksi. Hyväksytty ECR johtaa muutospyyntötilaukseen (ECO)(Rautelin, 2022). Muutospyyntötilausta käytetään suunnitteluprosessiin, jonka lopputuloksena havaittu muutostarve toteutetaan alkaen suunnittelusta loppuen tehtaalta lähtevään tuotteeseen päätyneeseen muutokseen. Jos kyseessä on riittävän pieni muutostarve, virheellisen tuotteen saanut asiakas voi saada korjauksen hankintaansa varaosatoimituksena, joka asennetaan asiakkaalle jo toimitettuun tuotteeseen.

Valmistavan teollisuuden yrityksille on tyypillistä, että ne keskittyvät pitkiä aikoja tietyn tuotekategorian tuotteiden valmistamiseen. Esimerkiksi AGCO Corporation ja siihen kuuluvat yritykset ovat valmistaneet yli sata vuotta moottoreita, ja vuosikymmeniä maataloustuotantolaitteita, kuten traktoreita. Valmistettavan tuotteen, kuten traktorin tyypin pysyessä samana, ajan saatossa siitä tehdään lukuisia eri versioita. Vaikka traktorin toimintaperiaate pysyisi pitkään pohjimmiltaan samana, tekniikan kehittyessä ja erilaisten käyttötarkoitusten syntyessä tuotteesta syntyy tuoteperhe, ja tuoteperheen osille syntyy uusia sukupolvia. Tuoteperheiden ja tuotesukupolvien välillä tuotteissa on yhtäläisyyksiä, mutta myös eriäväisyyksiä. Tietyn tuotesarjan runko voi pysyä samana useita sukupolvia, mutta muun muassa moottori vaihtuu. Tuotteen valmistamisen kannalta on oleellista tietää, mitä versioita kustakin järjestelmästä (kuva 1) ja tarkemmin kustakin komponentista kyseiseen tuotteeseen tarvitaan. Näitä informaationhallintahaasteita ratkaisemaan PLM-järjestelmä tarjoaa useita auttavia logiikoita.

PLM-järjestelmän sisäisessä logiikassa tuote koostuu kokoonpanoista, jotka koostuvat osakokoonpanoista. Osakokoonpanot sisältävät nimikkeitä, jotka edustavat muun muassa osia. Jokaisella osalla, kokoonpanolla ja tuotteella on revisio, joka kertoo, monesko versio se on kyseisestä nimikkeestä. Sovelia Core PLM käyttöliittymä kertoo myös käyttäjälle, katseleeko hän nimikkeen uusinta revisiota. Revision lisäksi tuotteen elinkaaren hallitsemiseksi käytetään elinkaaren tiloja (taulukko 1). Elinkaaren tilat kertovat katsojalle, millaisessa elinkaarensa vaiheessa nimike on. Suunnittelijalle merkityksellisimpiä ovat tilat ennen tilaa *Tuotannossa*. Myyjälle *Tuotannossa*-tilassa olevat nimikkeet ovat mielenkiintoisia. Varaosatoimituksista vastaavalle *Myynninjälkeisessä käytössä*-tilassa olevat nimikkeet ovat keskeisiä. Kaikille käyttäjäryhmille on myös oleellista tietoa, jos nimike on vanhentunut, ja sitä ei voi enää käyttää. Tällaisen nimikkeen tila on nimeltään *Käytöstä poistettu.*

Taulukko 1. Elinkaaren tilat (Ray, 2025)

Tila	Kuvaus
Suunnittelussa	Suunnittelutyö on käynnissä
Suunnittelun katselmus	Suunnittelu on arvioitavana
Suunnittelu valmis	Suunnittelu on valmis ja odottaa hyväksyntää
Tuotannossa	Tuotannossa. Muutokset vaativat uuden revision
Myynninjälkeisessä käytössä	Ei enää valmisteta. Voidaan käyttää esim. varaosatoimituksiin
Käytöstä poistettu	Vanhentunut. Ei voi enää käyttää

2.4 Esimerkkitapauksia

Esimerkit ovat kyseisiä tuotteita toimittavien yritysten julkaisemia ja siten ne ajavat julkaisijansa intressejä. Siten esimerkkeihin tulee suhtautua kriittisesti, sillä referenssin julkaisevien tahojen intresseissä on näyttäytyä kaupallisesti positiivisessa valossa. Tapaukset tarjoavat kuitenkin lukijalleen informaatiota toteutetuista onnistuneista ratkaisuista, joten ne tarjoavat lukijalleen käyttökelpoista tietoa varauksin.

2.4.1 Tuotteen elinkaaren hallinnalla tehokkuutta Europressin räätälöityihin jätepuristimiin

Jätepuristimia valmistavan Europress-yrityksen case-esimerkki Sovelia Core PLM:n ja sen liitännäisen Sovelia Configuratorin käyttöönottoprojektista.

Monet asiakkaat tarvitsivat räätälöityjä ratkaisuja jätepuristimen koon, kapasiteetin ja toiminnallisuuksien suhteen. Räätälöityjen tuotteiden suunnittelu ja valmistus veivät paljon aikaa, mikä hidasti toimitusaikoja. Monimutkaiset ja manuaaliset räätälöintiprosessit lisäsivät suunnitteluvirheiden riskiä. Tuotteet ja rakenteet luotiin manuaalisesti ERP-järjestelmään, mikä teki prosesseista aikaa vievää. Hidas tiedonkulku suunnittelijoiden ja tuotannon välillä hidasti toimintaa. Yrityksellä oli tarve tehokkaalle tuotteen elinkaaren hallinnalle.

Sovelia Core PLM -järjestelmä optimoitiin yrityksen toimintoihin. Sovelia Configuratoria käytetään hallitsemaan asiakaskohtaisia tuotevariaatioita ja automatisoimaan suunnittelua. Näiden ratkaisujen avulla kaikki tuotetiedot ovat näkyvissä yhdessä järjestelmässä, mikä säästää aikaa ja vaivaa. Huoltotiimit eri maissa voivat tarkastella kuvia, rakenteita ja varaosatarpeita helposti.

Suunnittelukäytännöt ja laatu paranevat ja standardisoituvat, mikä vähentää inhimillisten virheiden riskiä. Toimitusajat nopeutuvat merkittävästi, ja räätälöityjen tuotteiden suunnittelu on nopeampaa Sovelia Configuratorin avulla. Suunnittelijoiden työaika vapautuu varsinaiseen suunnittelutyöhön, ja suunnitelmien tarkastelu ja hyväksyminen on helpottunut. Lisäksi tiedot ovat saatavilla verkkokäyttöliittymän kautta mobiililaitteilla, ja suunnitelmat voidaan automaattisesti tarkastella 3D-malleina. Kaikki tuotedokumentit ja tilaushistoriatiedot ovat sujuvasti saatavilla yhdessä paikassa. (Symetri, n.d.)

2.4.2 Tamturbon matka Excelistä ammattimaiseen tuotetiedon hallintaan

Suomalainen kompressorivalmistaja Tamturbo siirtyi Excelistä ammattimaiseen tuotetiedon hallintaan ottamalla käyttöön Aras Innovator -alustan. Vuonna 2010 perustettu yritys valmistaa täysin öljyttömiä kompressoreita ja tarjoaa myös Air-as-a-Service -liiketoimintamallia, jossa asiakas maksaa vain käyttämästään paineilmasta. Yrityksen kasvaessa Excel-pohjainen tiedonhallinta alkoi osoittaa puutteensa: tiedostoja kopioitiin, muutettiin ja tallennettiin uusina versioina ilman kunnollista versiohallintaa. Palvelinrakenteiden monimutkaistuessa ajantasaisen tiedon löytäminen vaikeutui merkittävästi.

Tamturbo valitsi Aras Innovator -alustan sen joustavuuden, helpon migraation ja käyttäjäystävällisyyden vuoksi. Käyttöönotto toteutettiin kolmessa vaiheessa vuosina 2018–2020 yhteistyössä Fulvisolin ja Focus PLM:n kanssa. Perustoiminnot otettiin käyttöön ensin, minkä jälkeen lisättiin liiketoiminnan kannalta kriittiset toiminnot ja lopuksi tehtiin järjestelmän parannuksia.

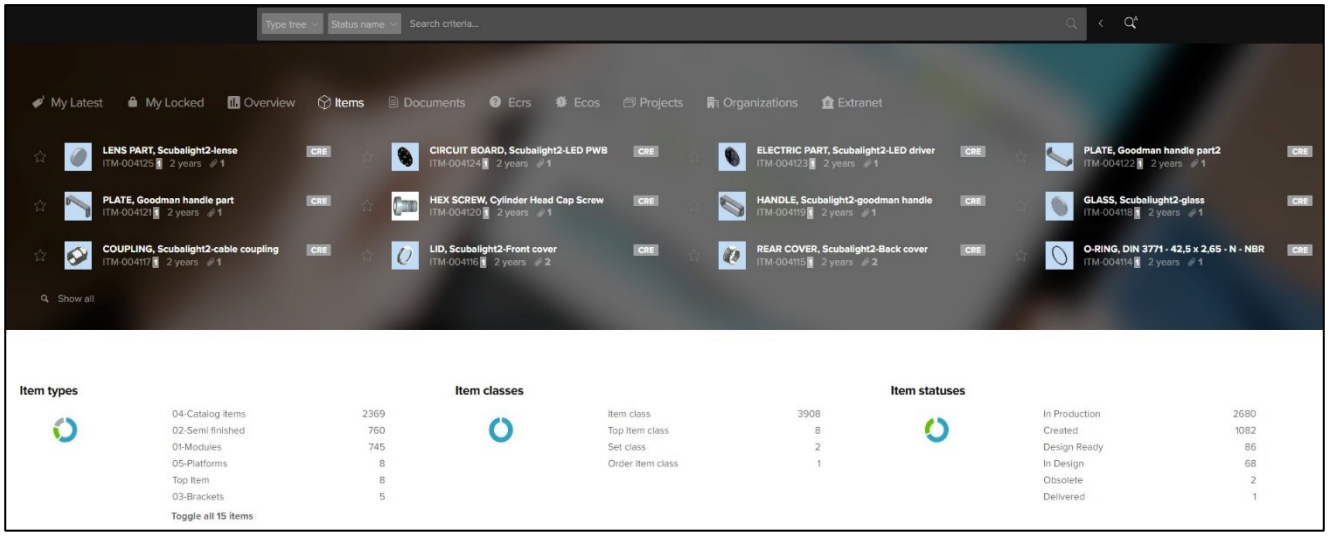
Nykyään lähes kaikki Tamturbon työntekijät käyttävät Aras Innovatoria. Oikean suunnittelutiedon etsiminen vei aiemmin tunteja tai jopa päiviä, mutta nyt se onnistuu muutamassa minuutissa. Tuotteiden visualisointi on mahdollista kaikille työntekijöille ilman CAD-ohjelmistoa, mikä helpottaa osastojen välistä yhteistyötä. Myynti ja huolto voivat näyttää asiakkaille kompressorit ja yksittäiset osat tarkemmin kuin ennen. Järjestelmä antaa yritykselle kokonaiskuvan kompressoreiden koko elinkaaresta ja tukee kestävään kehitykseen liittyvän tiedon, kuten hiilipäästöjen seurantaa ja mittaamista. (Aras, 2022)

2.5 Sovelia PLM lyhyesti

Sovelia Core PLM:n käyttöliittymässä käyttäjä näkee kotisivulla ensimmäisenä (kuva 5) listan viimeisimmistä itse muokkaamistaan nimikkeistä, hakukentän sekä tilastoja liittyen järjestelmän sisältämiin nimikkeisiin. Hakukentän vasemmalla puolella olevasta painikkeesta *Typetree* pääsee valikkoon, jossa näkyy järjestelmään määritelty tyyppipuu. Tyyppipuu on PLM järjestelmäkonfiguraation

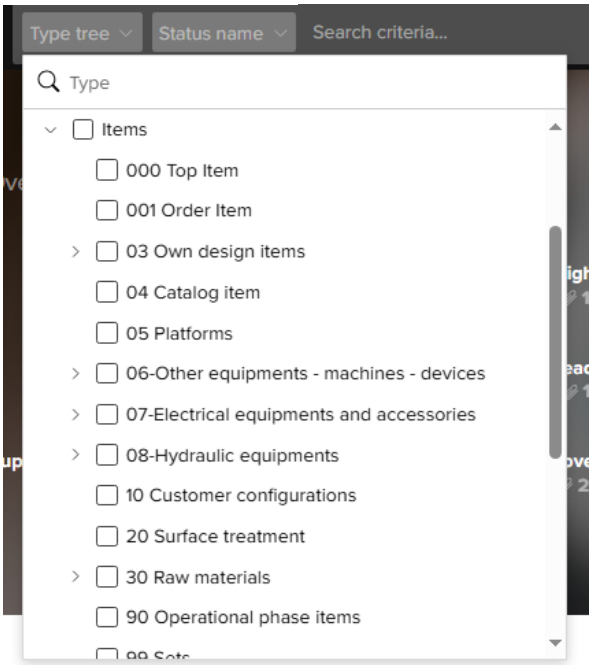
ydin. Jokaisella PLM:n käyttöön ottavalla yrityksellä on omanlaisensa hierarkia, jonka mukaan tuotteen valmistaminen järjestetään.

Kuva 5. Sovelia Core PLM kotisivu



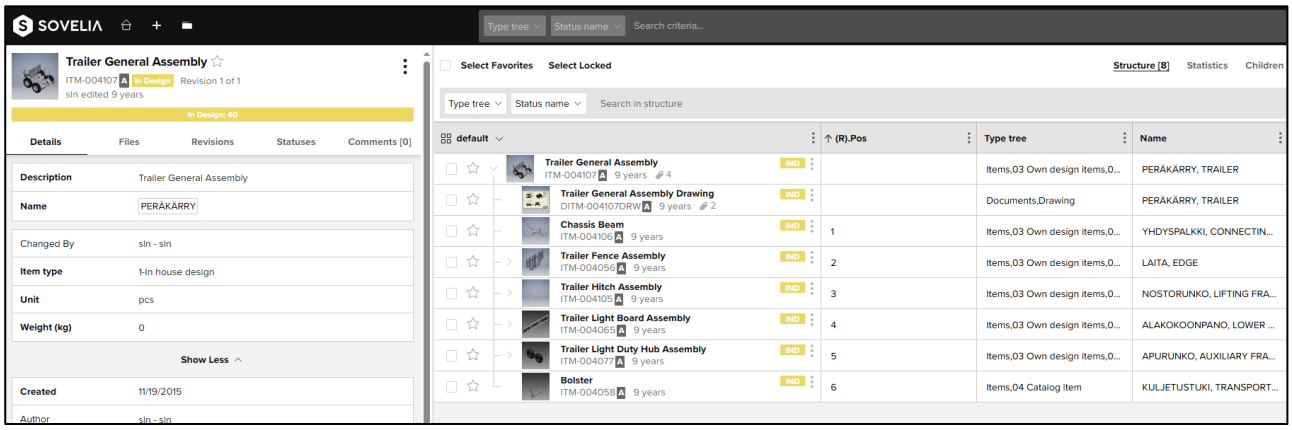
Tyyppipuu on Sovelia Core PLM:n tapa muodostaa tuo hierarkia. Se on puurakenne, jossa kaikki tuotteen valmistamiseen liittyvä tieto kootaan sellaiseksi loogiseksi kokonaisuudeksi, että tietoa pystytään hyödyntämään tehokkaasti. Esimerkkityyppipuussa (kuva 6) on esitetty tyyppipuuhierarkian Nimikkeet-haara (Items) jonka alla on kategorioita, kuten Itse suunnitellut nimikkeet, sähkökomponentit, hydrauliikkakomponentit, raaka-aineet.

Kuva 6. Tyyppipuu



Tyyppipuun kuvatessa nimikkeiden hierarkiaa, varsinaisessa suunnittelutyössä, jossa määritellään valmistettavan koneen rakenne, käytetään sen esittämiseen rakenteita (kuva 7). Kuvassa ylimmällä tasolla on koko konetta kuvaava nimike (Trailer General Assembly), ja sille alistettuna alemmilla riveillä ovat kaikki ne komponentit, joista kone muodostuu. Komponenttien lisäksi koneen rakenteessa on myös kaikki siihen liittyvät dokumentit, jotka voivat sisältää muun muassa piirustuksia, raportteja sekä tekstidokumentteja. Tekstidokumentit voivat kertoa esimerkiksi koneen toimintaperiaatteesta, tai sen valmistamisessa huomioitavista turvallisuuskäytänteistä.

Kuva 7. Rakenne



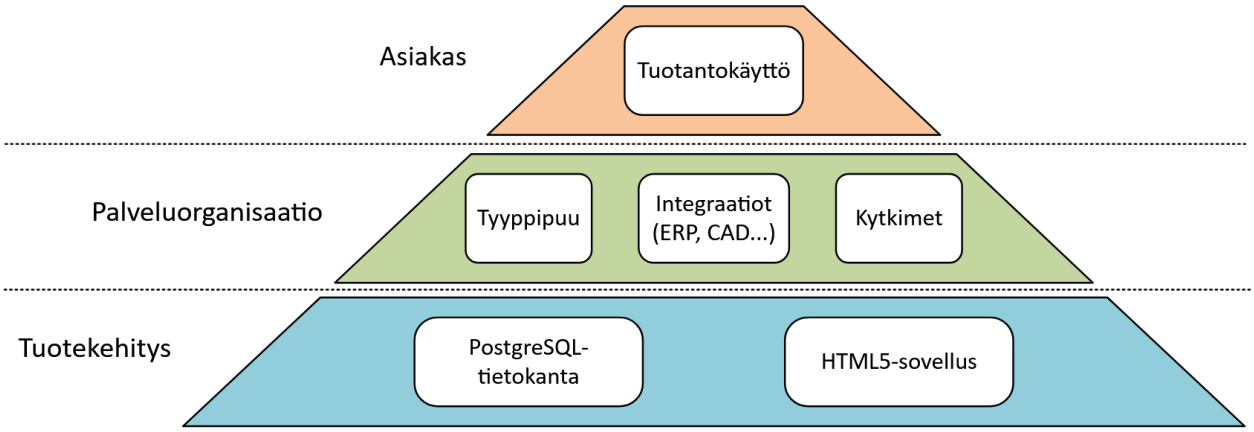
2.6 Muutosten syntyminen PLM-järjestelmässä

Sovelia Core PLM-järjestelmässä syntyy muutoksia kolmessa pääkategoriassa (kuva 8). Järjestelmän ydin, jonka Sovelia Core PLM tuotekehitysosasto tuottaa muodostuu PostgreSQL-tietokannasta ja HTML5-sovelluksesta. Sen päälle rakentuu järjestelmä, jota muokataan asiakkaan tarpeiden mukaan. Järjestelmäasennuksen mukana tulevien perustoimintojen lisäksi jokaisessa asiakaskohtaisessa asennuksessa palveluorganisaatio konfiguroi tarvittavat lisätoiminnot. Asiakasyrityksen valmistamasta tuotteesta riippuu, millainen tyyppipuu muodostetaan. Yrityksen käyttämät muut järjestelmät, kuten ERP ja CAD määrittävät sen, millaisia integraatioita järjestelmien välille täytyy rakentaa. Edellisessä luvussa avattu yrityksen historia vaikuttaa myös oleellisesti siihen, millaisia toimintoja esimerkiksi ERP:issä on jo käytössä. Jos sinne on ehditty rakentaa paljon sellaista toiminnallisuutta, jonka voisi tehokkaasti tehdä myös PLM:ssä, ei konfigurointityötä välttämättä tehdä uudelleen. Jos taas jo rakennettuihin toiminnallisuuksiin liittyy uusia kehitystarpeita, sen rakentaminen uudelleen PLM:ään voi ollakin kannattavaa. Sovelia Core PLM:n sisäisillä automaatiotyökaluilla (kytkimillä) ohjataan PLM:ssä monia erilaisia taustalla tehtäviä automatisoituja toimintoja. Esimerkiksi suunnittelijan todetessa suunnittelutyön valmiiksi (taulukko 1) hän valitsee Sovelia PLM Coressa suunnitellun nimikkeen tilan vaihdoksen tilasta *Suunnittelussa* tilaan *Suunnittelun katselmus*. Tässä tilanvaihdoksessa voi olla asetettuna kytkin, joka

ennen tilan vaihtoa varmistaa, että nimikkeelle on annettu kaikki vaaditut tiedot, kuten mitat ja materiaalit, joista nimike valmistetaan.

Järjestelmäasennuksen valmistuttua asiakas aloittaa järjestelmän käytön osana tuotantoprosessiaan. Eri käyttäjäryhmät suunnittelijoista tuotantotyöntekijöihin sekä ostajista myyjiin alkavat käyttää järjestelmää omissa käyttötarkoituksissaan eri tavoin. Eri käyttötavoista jää erilaisia jälkiä järjestelmään. Nimikkeiden perustamiset, muokkaukset ja tilamuutokset jättävät jälkiä järjestelmän lokitiedostoihin. Nimiketietojen lähettämiset CAD:ista PLM:ään ja PLM:stä ERP:iin jättävät jälkiä sekä PLM:n omiin lokitiedostoihin, että lähde- ja kohdejärjestelmiin. Integraatioilla itsellään voi olla omat lokitiedostot, joihin jää myös jälkiä. PLM-järjestelmään tallennetut tiedot sijaitsevat tarkalleen ottaen tietokannassa, joten muutokset tiedoissa näkyvät myös tietokannassa.

Kuva 8. Muutosten pääkategoriat



PLM-järjestelmäasennuksen lähtötilanne on käyttöönottoprojektissa tehty asennus, johon on tehty asiakkaan käyttötarkoituksen huomioivat sekä muut tarkemmat asiakkaan tarpeeseen räätälöidyt toiminnallisuudet. Käyttöönottoprojektin aikana asiakas tutustuu ja oppii PLM-järjestelmästä sekä sen käytöstä paljon uutta. Paljon syväoppimista tapahtuu käyttöönottoprojektin jälkeen, kun asiakas alkaa käyttää järjestelmää niissä päivittäisissä tehtävissä, joita varten järjestelmä hankittiin. Ymmärryksen ja osaamisen kehittyessä järjestelmän käyttö muovautuu, käyttöön syntyy asiakasyrityksen sisäisiä rutiineja, ja yksittäiset käyttäjät oppivat käyttämään järjestelmää omiin työskentelytapoihinsa sopivalla tavalla. Ymmärrys järjestelmästä voi avata paljon uusia ajatuksia sen tarjoamista mahdollisuuksista sekä synnyttää halua ottaa PLM-järjestelmässä käyttöön lisää toiminnallisuuksia. Muuttuvat, kehittyvät tarpeet muovaavat vaatimuksia itse järjestelmälle, ja ajan kuluessa asiakasyritykselle voi syntyä tarve tilata järjestelmätoimittajalta uusia toiminnallisuuksia tai muutoksia jo olemassa oleviin. Asiakaskohtaisesti nämä muutostarpeet voivat syntyä kuukausien tai vuosien päästä alkuperäisestä käyttöönotosta.

Järjestelmätoimittajan työpanosta vaativia muutoksia varten sen täytyy uusien muutosten tullessa ajankohtaiseksi päivittää omat tietonsa ajan tasalle asiakaskohtaisesta järjestelmäasennuksesta ja asiakkaan käyttämistä toiminnoista. Muutoksia asiakaskohtaiseen järjestelmään on ajan saatossa voinut syntyä esimerkiksi Sovelia PLM:n automaattisia rutiinitoimintoja käyttäviin kytkimiin, tyyppipuun haaroissa oleviin tietokenttiin, tietonäkymiin tai funktioihin.

2.7 Muutosten dokumentoinnin merkitys

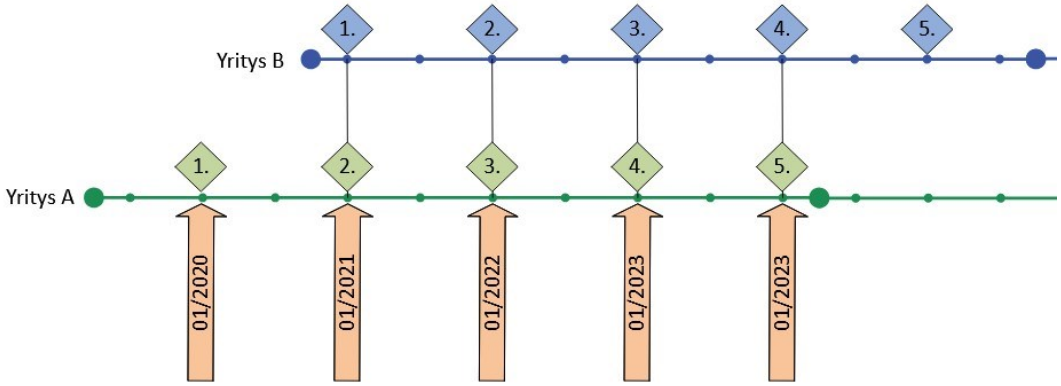
Hyvin suunniteltu ja toteutettu dokumentointi mahdollistaa järjestelmän käyttötapausten muuttuessa sen ymmärtämisen tehokkaasti. Toisinaan asiakasympäristöön tehtävien muutostöiden välillä voi kulua pitkä aika, jolloin osa edellisessä muutostyössä kertyneestä tiedosta hämärtyy tai vanhentuu uusien toimintojen käyttöönoton myötä. Dokumentointi mahdollistaa nopeamman järjestelmäasennuksen erityisominaisuuksien muistiin palauttamisen sekä sen ymmärtämisen. Ilman dokumentointia järjestelmäasennuksen kanssa aiemmin työskennellyt pystyy palauttamaan mieleen siitä aiemmin opitut asiat. Hyvä dokumentaatio voi kuitenkin nopeuttaa mieleen palauttamista. Dokumentointi auttaa uutta, vailla aiempaa kokemusta kyseisestä järjestelmäasennuksesta olevaa henkilöä ymmärryksen muodostamisessa.

Jos jonkin parametrin esiintyminen lokitiedostoissa jokaiselta työpäivältä vuodessa haluttaisiin laskea käsin, se veisi turhan paljon aikaa suhteessa siihen, että sen tiedon automaattisen dokumentoinnin toteuttaminen on mahdollista tehdä melko pienellä vaivalla. Käsin dokumentoinnin tekeminen on vaivalloista ja aikaa vievää, joten dokumentointi täytyy automatisoida niiltä osin, kuin se on tehokasta toteuttaa. Uusi automaattista dokumentointia tekevä toiminnallisuus tallentaisi valitut dokumentoitavat parametrien arvot esimerkiksi kerran vuorokaudessa. Parametrien arvoista koostettaisiin dokumentaatio, jonka avulla voidaan muodostaa kokonaiskuva järjestelmän käytön kehittymisestä esimerkiksi puolivuosittain tai kvartaaleittain. Eri asiakkuuksien konfiguraatioista dokumentoituja parametreja voitaisiin tarkastella keskenään, jolloin opittaisiin lisää siitä, miten erilaisissa käyttökohteissa PLM-järjestelmää käytetään eri tavoin.

Esimerkissä (kuva 9) on kaksi yritystä, jotka ovat ottaneet PLM-järjestelmän käyttöön eri aikoihin. Ensimmäinen automaattinen dokumentointi yritys A:lle tehdään 01/2020, toinen 01/2021 ja yritys B:lle ensimmäinen tehdään 01/2021. Seuraavat dokumentoinnit tehdään vastaavasti molemmille yrityksille aina seuraavassa tammikuussa. PLM-järjestelmään alkaa syntyä muutoksia, jotka näkyvät dokumentoinnissa heti järjestelmän käyttöönotosta alkaen (kuva 9). Osa muutoksista järjestelmissä ovat samankaltaisia eri yritysten järjestelmissä, mutta osa niistä ovat erilaisia. Dokumentoinnin vertailukelpoisuuden varmistaminen on tärkeää ottaa huomioon sitä suunniteltaessa, jotta kahden eri

yrityksen muutosten vertailusta voidaan saada hyödyllistä informaatiota. Vähintäänkin muutokset Sovelia Core PLM:n perustoiminnallisuuksissa, joita molemmat vertailun yritykset käyttävät tuottavat vertailukelpoista tietoa muutoksista. Toinen näkökulma kahden eri järjestelmäasennuksen dokumentaatioiden vertailuun on dokumentaatio tietyssä ajanhetkessä. Olettaen, että molemmilla yrityksillä on uusin Sovelia PLM Core versio asennettuna ajanhetkellä tammikuu 2021, jolloin yritys B:n ensimmäinen ja yritys A:n toinen dokumentointi on tapahtunut, näitä dokumentointeja voidaan verrata saman järjestelmäversion erilaisten käyttötapojen näkökulmasta. Kuten aiemmassa käsittelyssä on havaittu, Sovelia Core PLM -järjestelmän dokumentoinnille on merkittävä tarve. Kytkinten käyttömäärien tai käyttökertojen onnistumisten ja epäonnistumisten manuaalinen kerääminen olisi erittäin työlästä,

Kuva 9. Kahden yrityksen automatisoitu dokumentointi aikajanalla



varsinkin kun tietotekniikan kehitys tarjoaa mahdollisuuksia automatisoituun tiedonkeruuseen. Tämä viittaa siihen, että muutosten dokumentointiin tarvitaan tehokas työkalu, joka kykenee keräämään ja jäsentämään tiedot informatiiviseen muotoon.

2.8 Nykyiset muutosten dokumentointimenetelmät ja niiden rajoitukset

Nykyisin käytössä olevia muutosten dokumentointimenetelmiä ovat versionhallinta Mercurial, SQL-muutostentunnistustyökalu sekä asiakaskohtaiset kirjoitetut dokumentit (taulukko 3). Rajoituksia nykyisten dokumentointitapojen käytölle muodostavat esimerkiksi dokumenteista tiedon etsimisen vaikeus sekä se, ettei niissä ole dokumentoituna kaikkea sitä muutostietoa mitä tarvitaan. Se, että tietoa on tallennettu muistiin, ei tee siitä sellaisenaan käyttökelpoista dokumentaatiota. Dokumentaatio täytyy suunnitella parhaassa tapauksessa etukäteen niin, että siitä saa tarvittavat tiedot käyttöön, sekä se, että tarvittavat tiedot löytyvät vaivattomasti.

Varsinaisten käytössä olevien dokumentointimenetelmien lisäksi tietoa kerätään erilaisiin lokitiedostoihin sekä monitorointityökaluihin nähtäville, esimerkiksi Azure Monitoring-järjestelmään. Hajallaan oleva tieto lokitauluissa ja lyhyessä ajassa pois pyyhkiytyvät tiedot monitoroinnissa eivät täytä dokumentoinnin vaatimuksia. Monitorointi on lyhytaikaista tietojen tallentamista, dokumentointi sen sijaan pitkäaikaista. Ne palvelevat siis eri käyttötarkoituksia. Akuutin asiakkaan järjestelmässä olevan vikatilanteen paikantamiseksi suuri määrä erilaisia lokiin merkittäviä tietoja voi olla tarpeen. Sen sijaan dokumentointia varten tarvitaan sellaista tarkkaan harkittua tietoa, jota käsitellä pitkän ajan päästäkin.

Taulukko 2. Nykyiset dokumentointimenetelmät

Mercurial-versionhallinta
SQL-Muutostentunnistustyökalu
Asiakaskohtainen kirjoitettu tekstimuotoinen dokumentaatio

Monitorointi- ja dokumentointitiedon merkittävä ero on juuri tiedon tallentamisen tarkkuus. Usean vuoden aikana dokumentoitava muutostieto täytyy olla tarkkaan harkittua sen käyttötarkoituksen mukaan, jotta syntyvä tieto pysyy käyttökelpoisena, eikä sitä synny turhaan liikaa. Sekä monitorointi että automaattinen dokumentointi vievät järjestelmän resursseja, kuten tallennustilaa sekä suorituskykyä, joten on tärkeää ottaa huomioon, etteivät ne kuluta niitä liikaa. Tämä on tärkeää ottaa huomioon siksi, että PLM-järjestelmä on yritykselle tärkeä osa tuotantoprosesseja, joten palvelimen, jolta järjestelmää ajetaan, täytyy pystyä suoriutumaan ensisijaisesta käyttötarkoituksestaan hyvin. Monitorointi ja dokumentointi ovat toissijaisia resurssien käytön suhteen.

Hyvä järjestelmäasennuksen dokumentointi vaatii, että siihen tallennettava tieto tallennetaan selkeässä, vertailukelpoisessa muodossa sellaiseen sijaintiin, josta tieto on saatavissa sitä tarvitseville osapuolille pidemmänkin ajan kuluttua. Nykyiset dokumentointimenetelmät eivät kirjaa ylös kaikkia tarvittavia tietoja riittävällä tarkkuudella. Esimerkiksi kytkimien ajoa koskeva dokumentointi on usein hyvin rajallinen. Jos kytkimien käytöstä haluttaisiin tietoa, se täytyisi kaivaa käsin jokaisena päivänä syntyvästä lokitiedostosta erikseen. Jokainen lokitiedosto on helposti tuhansia rivejä pitkä. Pelkästään kuukauden

lokitiedostojen lukeminen käsin voisi käsittää satatuhatta riviä tekstiä. Sen avaaminen ja selaaminen käsin sanan esiintymismääriä mittaavan toiminnon avulla tekstieditorissa ei ole mielekästä. Siksi tarvitaan tehokkaampi ratkaisu automaattiseen dokumentointiin.

3 Tutkimuksen toteutus

3.1 Tutkimusmenetelmät ja lähestymistapa

Tässä tutkimuksessa käytetty tutkimuslähestymistapa koskien dokumentointia Sovelia Core PLM:n ulkopuolella voidaan luonnehtia laadulliseksi, tutkivaksi kirjallisuuskatsaukseksi. Tavoitteena oli tutkia ja soveltaa olemassa olevaa tietoa ja parhaita käytäntöjä automaattisen tiedonkeruun alalla tietojärjestelmistä. Tutkimus aloitettiin etsimällä tietoa tiedonkeräysmenetelmistä, dokumentointimenetelmistä, datatyypeistä sekä tiedon hyödyntämismenetelmistä tietojärjestelmissä. Seuraavaksi tarkennettiin tiedon keruuta koskemaan erityisesti yritysten liiketoiminnan kannalta merkittäviin liiketoimintajärjestelmiin liittyvään tiedon käsittelyyn. Tutkimuksen edetessä valittiin keskittyä kolmeen avainalueeseen: parametrien valinta, tiedonkeruumenetelmät ja tiedontallennusratkaisut. Suurin osa opinnäytetyössä käytetyistä lähteestä ovat erilaisten liiketoimintajärjestelmien ympärillä työtä tekevien yritysten tuottamia ja julkaisemia artikkeleja, joten niiden arvioinnissa tulee olla kriittinen.

Lähteinä käytettyjä kirjoja *Data Science for Business* sekä *The Data Warehouse Toolkit* voidaan pitää muita lähteitä luotettavampina, sillä ne ovat alansa tunnustettujen asiantuntijoiden kirjoittamia sekä uskottavien kustantajien julkaisemia. Yritysten tuottamien artikkelien julkaisun motivaationa voidaan pitää yleisesti lukijalle tietoisuuden lisäämistä yrityksen itsensä tuottamista palveluista, järjestelmistä tai muista ratkaisuista. Siten kukin yrityksen tuottama sisältö saattaa olla kallellaan omaa tuotetta kohti, mitä tulee suosittelussa erilaisten ratkaisuiden käyttöön. Yritysten tuottamat artikkelit, joita tässä opinnäytetyössä käytetään eivät ole tieteellisiä artikkeleita. Siten ne eivät edusta luotettavuudeltaan korkeinta tasoa. Perustuen kirjallisuuskatsaukseen, tutkimusmenetelmää laajennettiin sisältämään käytännön toteutusvaihe, joka hyödyntää tekijän työkokemusta Sovelia Core PLM-järjestelmän parissa. Tämä vaihe keskittyy kehittämään ratkaisua automaattiseen tiedonkeruuseen ja dokumentointiin. Toteutus koostuu kahdesta pääkomponentista:

1. Lokianalysaattori:
   Kehitettiin mukautettu Python-ohjelma, joka poimii tiettyä dataa PLM-järjestelmän tuottamista lokitiedostoista ja tallentaa sen JSON-tiedostoon. Tämä ohjelma havainnollistaa kirjallisuuskatsauksessa tunnistettujen tiedonkeruumenetelmien käytännön soveltamista, keskittyen lokitiedostojen analysointiin keskeisenä tiedonkeruutekniikkana.

2. Visualisointityökalu:
   Luotiin täydentävä JavaScript-ohjelma tuottamaan pylväsdiagrammi lokianalysoijan JSON-tulosten perusteella. Tällä työkalulla osoitetaan tiedon esittämisen ja analyysin merkitystä automaattisten datankeruujärjestelmien yhteydessä.

3.2 Tiedon kerääminen muissa viitekehyksissä

Tiedon saatavuus, tyyppi sekä sen käyttötarve määrittävät, millaisia parametreja valitaan kerättäväksi. Liiketoiminnallista etua tavoittelevassa tiedon keräämisessä toimiala vaikuttaa keskeisesti siihen, millaista tietoa on saatavilla, sekä millaista tietoa tarvitaan. Liiketoiminnan kehittämisen kohde määrittää edelleen tarkemmin sitä, millaista tietoa tarvitaan. Valmistusälyratkaisuja (Manufacturing intelligence) tuottavan Factbirdin mukaan valmistavan teollisuuden yrityksen datan keräämiseen tuotannosta on viisi vaihetta (kuva 10).

Factbirdin listaamat datan keräämisen lähtökohdat liittyvät tuottavan teollisuuden yrityksen datan keräämiseen fyysisestä tuotantoympäristöstä, mutta ne soveltuvat myös vastaavan tuotantotilanteen synnyttämien datavirtojen seuraamiseen tietojärjestelmissä, joiden avulla fyysistä tuotantoa hallitaan. Ensin täytyy tietää, miksi dataa ollaan keräämässä. Syy voisi olla esimerkiksi halu tietää, kuinka usein erilaiset tuotantolaitoksessa olevien sensorien välittämä tiedonsiirto epäonnistuu. Toisena täytyy selvittää, millaisesta tiedon siirron epäonnistumiseen liittyvistä tiedoista olisi eniten hyötyä prosessin kehittämisen kannalta. Kolmantena täytyy selvittää, missä osaprosessissa tässä esimerkkitapauksessa vian korjaamisesta olisi eniten hyötyä. Neljäntenä täytyy kehittää ratkaisu, jolla päästään käsiksi oleellisimpaan dataan. Viidentenä ja viimeisenä vaiheena täytyy ymmärtää ympäristöä, josta dataa ollaan keräämässä. (Bosson, 2023)

Nykyaikaisissa organisaatioissa tehokas ja luotettava tiedonhallinta on kriittinen menestystekijä. Tietojärjestelmien monimutkaistuessa ja datalähteiden lisääntyessä on syntynyt tarve järjestelmälliselle prosessille, jolla hajallaan oleva tieto saadaan koottua, yhdenmukaistettua ja saataville helposti. Tällaisen järjestelmällisen lähestymistavan puuttuminen johtaa usein epäjohdonmukaisiin tietokokonaisuuksiin, epäluotettaviin analyyseihin ja viime kädessä virheellisiin liiketoimintapäätöksiin. Päätöksentekijät tarvitsevat laadukasta, ajantasaista ja yhtenäistä tietoa, jotta he voivat tehdä perusteltuja päätöksiä muuttuvassa liiketoimintaympäristössä.

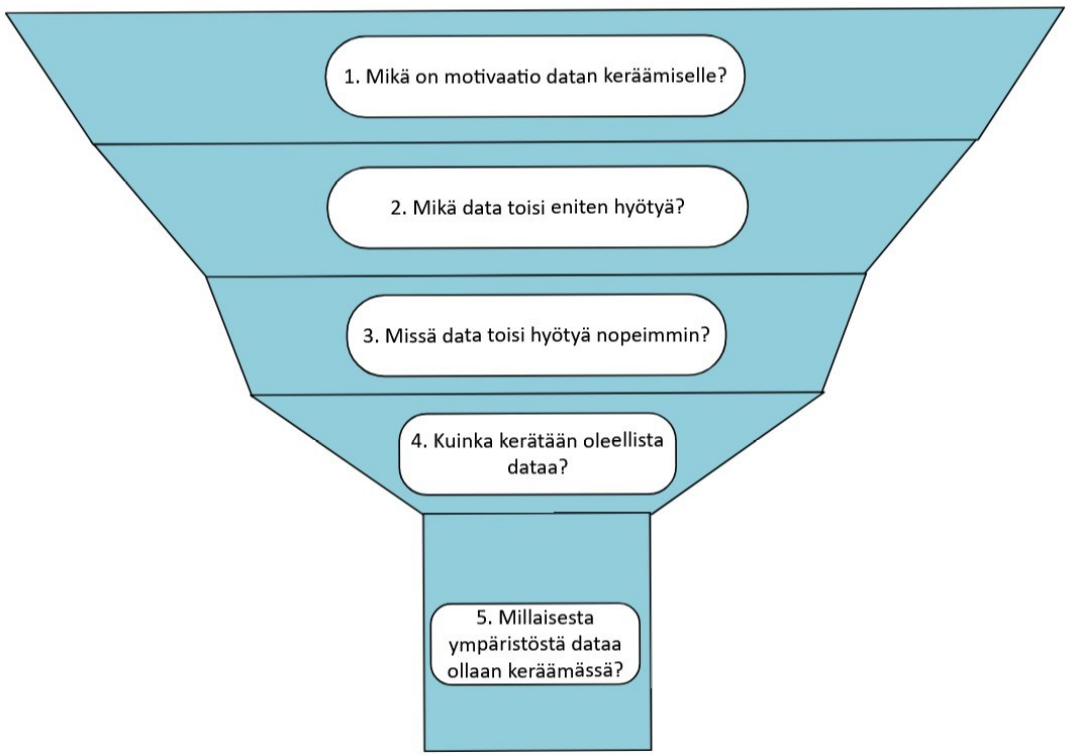
Kirjassa The Data Warehouse Toolkit esitellään ETL-prosessi, jonka vaiheita tässä luvussa käsiteltävät tiedon keräämisen vaiheet mukailevat. ETL-prosessin vaiheet ovat datan purkaminen, muuntaminen ja

lataaminen. Purkuvaiheessa data puretaan input-järjestelmästä, muuntamisvaiheessa se muutetaan haluttuun muotoon ja lopuksi lataamisvaiheessa se ladataan output-järjestelmään datan varastoimiseksi.

Kun kysytään, mikä on paras tapa suunnitella ja rakentaa ETL-järjestelmä, monet suunnittelijat vastaavat: "No, se riippuu." Se riippuu lähteestä, tietojen rajoituksista, käytettävissä olevista komentosarjakielistä ja ETL-työkaluista, henkilöstön taidoista sekä BI-työkaluista. Kuitenkin "se riippuu" -vastaus on vaarallinen, koska siitä voi tulla tekosyy jäsentymättömän lähestymistavan omaksumiselle ETL-järjestelmän kehittämisessä. Tämä pahimmassa tapauksessa johtaa sekavaan spagettisotkuun, joka koostuu taulukoista, moduuleista, prosesseista, komentosarjoista, triggereistä, hälytyksistä ja työaikatauluista. (Kimball, Ross, 2013, s. 443)

Vaikka Factbirdin esittämät vaiheet tarjoavat selkeän rakenteen datan keräämiselle, on tärkeää huomioida, että jokainen organisaatio kohtaa omat ainutlaatuiset haasteensa. ETL-prosessin parhaat käytännöt voivat auttaa näiden haasteiden hallinnassa ja varmistaa, että kerätty data tukee liiketoiminnan tavoitteita. On myös syytä pohtia, miten tiedonkeruujärjestelmät voivat mukautua nopeasti muuttuvaan liiketoimintaympäristöön ja teknologisiin innovaatioihin, kuten teollisen IoT:n ja tekoälyn hyödyntämiseen. Näiden teknologioiden integrointi voi tuoda uusia mahdollisuuksia datan analysointiin ja päätöksenteon tukemiseen, mutta samalla ne korostavat tarvetta systemaattiselle lähestymistavalle tiedonkeruussa.

Kuva 10. Datan keräämisen lähtökohdat



3.2.1 Kerättävien parametrien valinta

Tiedonkeruu ja analysointi ovat keskeisiä prosesseja nykypäivän dataohjautuvassa liiketoiminnassa. Yrityksillä on käytössään erilaisia työkaluja ja menetelmiä tämän toteuttamiseen, kuten esimerkiksi avoimen lähdekoodin tiedonkeruukirjastoja verkkosivujen analysointiin. Vaikka tässä tutkimuksessa keskitytään ensisijaisesti Sovelia Core PLM -järjestelmän parametrien valintaan, on hyödyllistä tarkastella tiedonkeruun periaatteita ja esimerkkejä laajemmassa kontekstissa. Tämä auttaa ymmärtämään parametrien valinnan merkitystä osana laajempaa tiedonhallinnan ja -analysoinnin kokonaisuutta.

Kerättävien parametrien valinta on keskeinen vaihe automaattisessa tiedonkeruussa, sillä se määrittää saatavan tiedon laadun ja hyödyllisyyden. Firecrawl on erityisesti julkisista verkkosivuista tiedon keräämiseen kehitetty avoimen lähdekoodin python-kirjasto, jota yritykset voivat käyttää tiedon keräämiseen ja tallentamiseen. Tiedonlouhinta nojaa tehokkaaseen datan keräämiseen, sen tallentamiseen ja sen analysointiin (Twin, 2024). Artikkelin mukaan tiedonlouhinta sisältää suuren tietomäärän tutkimista ja analysointia hyödyllisten datakuvioiden (data pattern) sekä trendien löytämiseksi. Sitä käytetään esimerkiksi luottoriskin hallintaan, petosten tunnistukseen sekä roskapostin suodattamiseen. Tiedonlouhinnan vaiheet voidaan jakaa neljään vaiheeseen (taulukko 3).

Taulukko 3. Tiedonlouhinnan vaiheet

Data kerätään ja syötetään datavarastoon paikalliselle palvelimelle tai pilvipalveluun
Yritysanalyytikot, johtoryhmät ja tietotekniikan ammattilaiset pääsevät käsiksi tietoihin ja päättävät, miten he haluavat järjestää ne.
Mukautettu sovellusohjelmisto lajittelee ja järjestää tiedot.
Loppukäyttäjä esittää tiedot helposti jaettavassa muodossa, kuten kaaviona tai taulukkona.

Omaa tuotettaan valmistavat yritykset voivat hyödyntää tiedonlouhintaa esimerkiksi oman toimintansa pullonkauloja, eli kokonaisprosessia hidastavien osaprosessien löytämiseen. Tämän tiedon avulla yritys voi tehostaa toimintaansa, kun se pystyy kohdentamaan kehitysresussit tärkeimpiin kohteisiin. Jos yritys pystyy keräämään tietoa tietojärjestelmästä, jota se toimittaa asiakkailleen, se voi päätellä asiakkaan todellisesta käytöstä monia eri asioita. Datan täysimääräinen hyödyntäminen vaatii tutkimusta ja analyysiä, joiden perusteella päätellään, mitä dataa tarkalleen ottaen tarvitaan, ja mihin käyttötarkoituksiin sitä aiotaan hyödyntää. (Tuychiev, 2025)

(1) Tiedonlouhinta kaavojen löytämiseksi ja mallien rakentamiseksi eroavat (2) tiedonlouhinnan tulosten hyödyntämisestä. Opiskelijat sekoittavat usein nämä kaksi prosessia datatieteissä, ja johtajat tekevät saman virheen keskustellessaan liiketoiminta-analytiikasta. Vaikka tiedonlouhinnan tulosten hyödyntämisen tulisi ohjata ja informoida itse tiedonlouhintaprosessia, nämä kaksi vaihetta on kuitenkin pidettävä selkeästi erillään toisistaan. (Provost, Fawcett, 2013, s. 25)

Sovelia Core PLM -järjestelmän parametrien valinnassa onkin tärkeää erottaa itse tiedonkeruun prosessi kerätyn tiedon hyödyntämisestä. Parametrien valinta tulee perustua sekä nykyisiin tiedossa oleviin tietotarpeisiin että mahdollisuuteen löytää odottamattomia kaavoja ja trendejä järjestelmän käytössä. Tässä tutkimuksessa huomio kiinnittyy erityisesti parametreihin, jotka kuvaavat järjestelmän käyttötapojen muutoksia ajan kuluessa sekä parametreihin, joiden avulla voidaan tunnistaa järjestelmän hyödyntämättömiä ominaisuuksia. Kerättävien parametrien valinnassa on lisäksi tasapainoteltava riittävän kattavuuden ja käytännön tehokkuuden välillä – kaikkea mahdollista tietoa ei ole järkevää kerätä, vaan on keskityttävä niihin datapisteisiin, jotka tuottavat suurimman lisäarvon palveluorganisaatiolle ja asiakkaalle. Tämä edellyttää sekä teknisen PLM-järjestelmän rakenteen ymmärrystä että näkemystä siitä, mitkä muutokset järjestelmässä ovat liiketoiminnallisesti merkittäviä.

3.2.2 Tiedon kerääminen

Tietoa voi kerätä monilla eri keinoilla, joista osa on tehokkaampia kuin toiset, ja eri menetelmät sopivat paremmin tiettyihin käyttötarkoituksiin kuin toiset. Tietojärjestelmästä tietoa kerättäessä käyttökelpoisia tiedonkeräysmenetelmiä ovat esimerkiksi API-integraatiolla sekä lokitiedostoista tiedon kerääminen (Airbyte, 2024). Tiedon kerääminen tulee toteuttaa niin, ettei se vaikuta PLM-järjestelmän suorituskykyyn oleellisesti. Järjestelmä voi kuormittua ajoittain varsinaisessa käytössä tapahtuvien hetkellisten kuormituspiikkien takia. Uusien kuormituspiikkejä aiheuttavien toimintojen lisäämistä järjestelmään tulee välttää. Automaattista dokumentointia varten datan kerääminen ei ole kiireellinen operaatio, joten se voidaan mahdollisuuksien mukaan ajoittaa vuorokaudessa sellaiseen ajankohtaan, jolloin järjestelmän kuorma on muutoin vähäinen.

3.2.3 Tiedon tallentaminen

Tiedon varastointiin on olemassa kaksi pääkategoriaa: paikallinen tietovarasto sekä pilvessä sijaitseva tietovarasto. Varastointimenetelmä voi olla myös hybridi edellä mainituista. Tietoa voi tallentaa osittain pilveen ja osittain paikalliseen tietovarastoon tai, esimerkiksi tiedon arkaluonteisuuden perusteella vähemmän salaisen tiedon pilveen ja salaisen paikalliseen varastoon, jonka tietoturvallisuus on paremmin oman organisaation käissä. (Hashemi-Pour, n.d.) Tiedon tallentamisen menetelmän valintaan vaikuttaa tiedon laadun ja määrän lisäksi fyysinen sijainti. Jos tuotetieto syntyy ja sitä käytetään fyysisesti samassa sijainnissa, paikallinen tietovarasto voi olla hyvä valinta. Sen sijaan, jos yrityksellä on monta toimipistettä maantieteellisesti kaukana toisistaan, pilveen tiedon varastoinnista saadaan hyötyä, sillä se sujuvoittaa eri toimipisteissä syntyvän tiedon käsittelyä toisilla toimipisteillä. Suurten palveluntarjoajien pilvialustojen, kuten AWS ja Azure eduksi verrattuna paikalliseen tietovarastoon kuuluvat vakaat ja nopeat tietoliikenneyhteydet sijainnista riippumatta. Eri toimipisteiden tuotetiedon käytön edut pilviratkaisulla verrattuna paikalliseen tietovarastoon korostuu, jos toimipisteiden välillä on pitkä välimatka.

3.3 Testausympäristön määrittely

Testausympäristö on konfiguroitu Windows-pohjaiselle PC:lle ja sisältää Sovelia Core PLM 24.2 asennuksen, joka koostuu PostgreSQL-tietokannasta taustajärjestelmänä sekä HTML5-pohjaisesta käyttöliittymästä. Testissä käytetään järjestelmää siten, että kytkin, josta lokitiedostoon tallennettavia tietoja käsitellään, aktivoituu. Kytkimet eli triggerit ovat pieniä automaatiokomponentteja PLM-järjestelmän logiikassa, joiden toiminnan seuraamisen ympärille testi keskittyy.

Testausprosessin vaiheet ovat:

1. Ennalta määriteltyjen testitapausten suorittaminen PLM-järjestelmässä
2. Järjestelmälokien luominen, jotka sisältävät triggerien suoritusdataa
3. Erityisen triggeriin liittyvän datan erottaminen raakadatasta
4. Poimitun datan kerääminen ja jäsennelty tallentaminen
5. Kootun datan visualisointi graafisten esitysten avulla

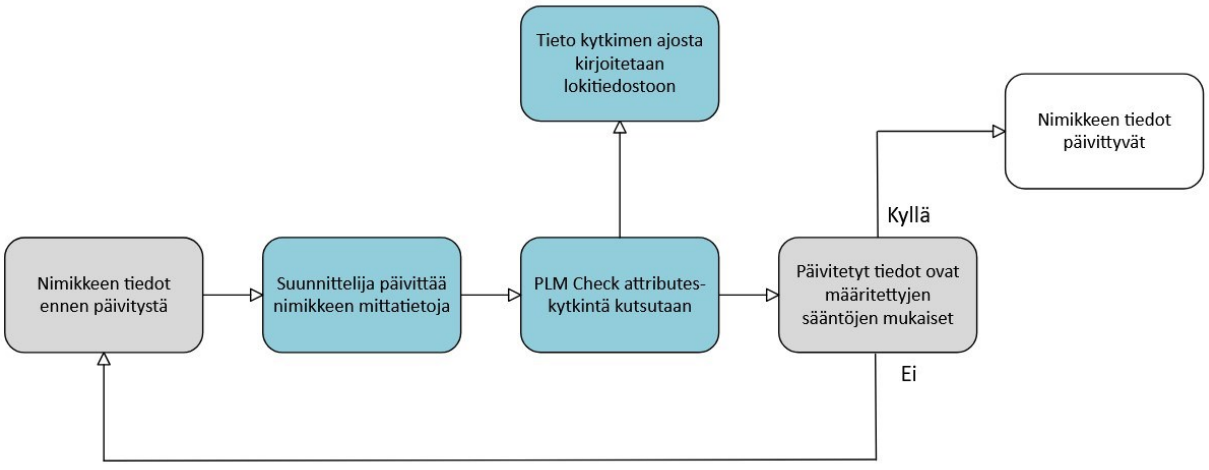
Käytettävä testiympäristö mahdollistaa kytkimen systemaattisen tarkkailun. Visualisoinnit selkeyttävät kytkimen käyttäytymistä ja tukevat automaatiomekanismien syvällisempää analyysiä.

3.4 Testattavat ominaisuudet ja mittarit

Yksi mahdollinen tapa toteuttaa automaattinen dokumentointi on seurata Sovelia PLM Coren kirjoittamaa lokitiedostoa (stdout.log) ja tallentaa siitä tarvittavaa tietoa myöhempää esitystä varten sopivaan muotoon. Lokitiedostoon tallennetaan esimerkiksi kytkinten kutsuja. Joka päivä syntyy yksi lokitiedosto, jossa on tallennettuna kaikki sen vuorokauden aikana tapahtuneet operaatiot tietyllä tarkkuudella kirjattuna. Järjestelmässä tapahtuu paljon erilaisia toimintoja, jotka tuottavat erilaisia jälkiä (kuva 11).

Tässä opinnäytetyössä käytetään esimerkkinä muutoksia PLM check attributes-kytkimen kutsumäärissä. Joka kerta kun kytkin ajetaan, siitä jää jälki lokitiedostoon (kuva 11). PLM Check Attributes-kytkin aktivoituu joka kerta, kun käyttäjä muuttaa nimikkeen tietoja ja tallentaa ne. Kytkin tarkistaa, että nimikkeelle annetut päivitetyt tiedot noudattavat asetettuja sääntöjä. Jos tiedot ovat sääntöjen mukaiset, muutos astuu voimaan. Jos tiedot ovat sääntöjen vastaiset, käyttäjälle tulostetaan ilmoitus sääntöjen vastaisesta tietojen muutoksesta, eikä yritetty tietojen muutos asetu voimaan.

Kuva 11. Kytkimen ajosta syntyvä jälki lokitiedostoon



PLM Check Attributes-nimisestä kytkimestä voi jäädä lokitiedostoon esimerkiksi kuvan 12 mukainen rivi. Rivi sisältää aikaleimat millisekunteina sekä päivämääränä ja tunteina, minuutteina ja sekunteina. Sen jälkeen rivillä on ilmoitettuna lokin kategoria, tässä tapauksessa SQL. Sen jälkeen on tietoa yhteyden tyypistä ja lopuksi funktiokutsu. Lokitiedostoon tallennetaan paljon erilaista informaatiota, ja tässä

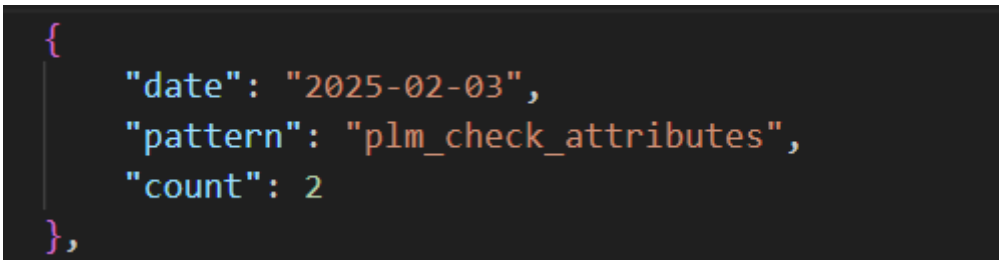
esimerkissä huomioidaan vain hyvin pieni osuus siitä. Kytkimien jälkeensä jättämiä tietoja voidaan päivittää tarkemmiksi sen mukaan, kun tarpeita tiedon keräämiselle keksitään lisää.

Kuva 12. stdout.log jälki kytkimen ajosta

Esimerkkitapauksessa lasketaan yksinkertaistamisen vuoksi ainoastaan kytkinkutsun esiintymismääriä, mutta varsinaiseen tuotantokäyttöön tullessaan ohjelma voitaisiin määrittää dokumentoimaan tarkempia, yksityiskohtaisempia tietoja kytkimen sekä muiden järjestelmän toimintojen tapahtumista.

Patterncount.py ohjelma lukee stdout.log-tiedoston ja laskee yhteen niiden rivien määrän, joilla esiintyy merkkijono **plm_check_attributes**. Tieto tallennetaan pattern_data.json-tiedostoon, jonka objektiin merkitään päivämäärä, jolloin ohjelma on ajettu, kyseessä oleva merkkijono sekä niiden esiintymismäärä (kuva 13). Pitkäaikaista tietojenkeräystä varten JSON-tiedostoja voidaan yhdistellä niin, että yhdessä tiedostossa on koko tarkasteluajan tiedot. Pidempiaikaisen tiedon seuraamisen kannalta yksikkö tapahtumaa per vuorokausi on todennäköisesti liian pieni. Todellisuudessa tapahtumia per viikko tai per kuukausi on lähempänä kiinnostavaa tarkastelutarkkuutta.

Kuva 13. Tallennettu tieto JSON-objektissa



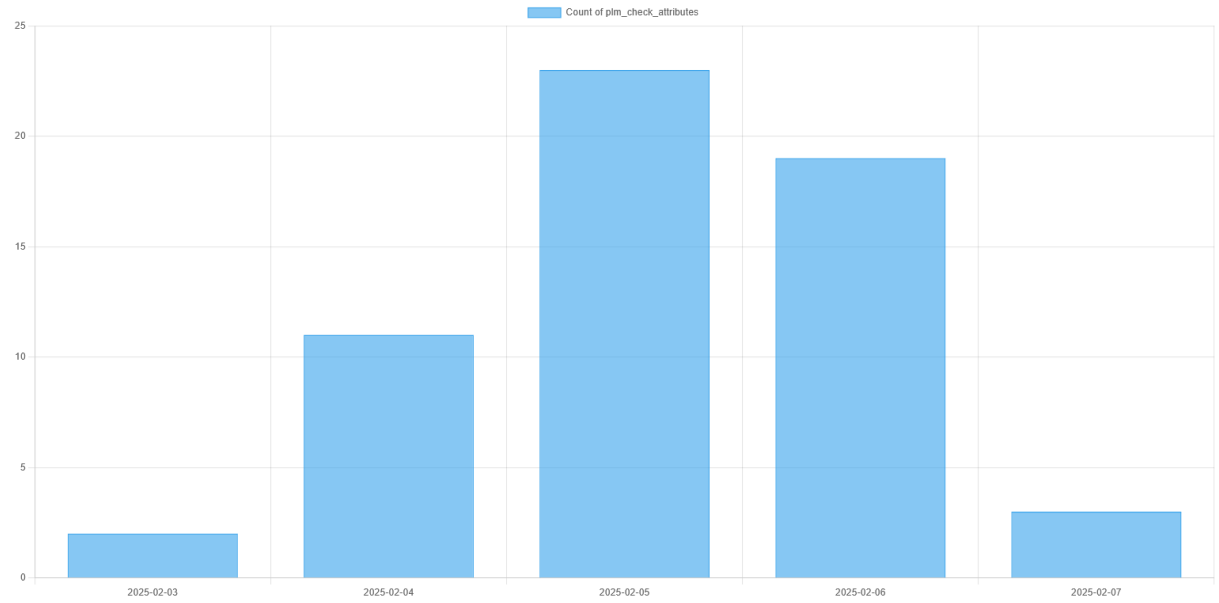
Python-ohjelma suorittaa seuraavat vaiheet:

1. Se ottaa lokitiedoston polun komentoriviparametrina tai käyttää oletuslokitiedostoa, jos polkua ei anneta
2. Se poimii ensimmäisen päivämäärän, joka löytyy lokitiedostosta muodossa VVVV-KK-PP
3. Se laskee, kuinka monessa rivissä lokitiedostossa on tietty merkkijono (testitapauksessa merkkijono "plm_check_attributes")
4. Se tallentaa tämän tiedon JSON-tiedostoon (pattern_data.json), lisäten uuden objektin, joka sisältää
    a. Lokitiedostosta poimitun päivämäärän
    b. Haetun merkkijonon
    c. Rivien määrän, jossa kyseinen merkkijono esiintyy

Skripti on konfiguroitu toimimaan UTF-8-koodattujen lokitiedostojen kanssa ja se käyttää säännöllisiä lausekkeita löytääkseen päivämäärät ja kuviot.

JSON-tiedostoon voidaan tallentaa sopivaksi katsottu määrä tietoja, esimerkiksi tiedot kaikilta päiviltä yhden kuukauden aikana. Tiedostoon tallennettu tieto esitetään index.html-tiedostoon toteutetulla logiikalla siten, että Chart.js-kirjaston avulla piirretään pylväsdiagrammi (kuva 14).

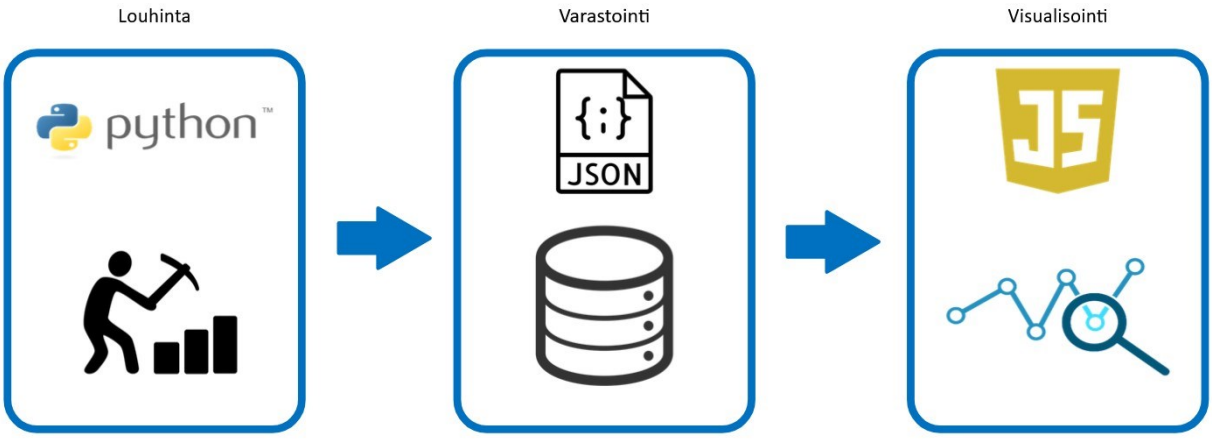
Kuva 14. JSON-datan esitys



Käytännön ratkaisun päävaiheet (kuva 15) ovat:

1. Tiedonlouhinta Python-ohjelmalla (patterncounting.py)
2. Tiedon varastointi JSON-formaattiin (pattern_data.json)
3. Tiedon visualisointi JavaScript-kirjaston avulla (index.html)

Kuva 15. Ratkaisun päävaiheet



3.5 Rajaukset ja haasteet

Tutkimuksessa keskitytään Sovelia Core PLM-järjestelmän automaattisen dokumentointiratkaisun kehittämiseen. Se ei ota kantaa järjestelmän käyttöliittymään tai käyttäjäkohtaisiin muutoksiin. Tutkimuksessa ei myöskään keskitytä tietoturvaan. Tutkimuksen rajoitteisiin sisältyvät tutkittavien parametrien määrä, tietokannassa tapahtuvien tärkeimpien muutosten tunnistaminen, automatisoidun dokumentoinnin kattavuuden varmistaminen sekä kerätyn datan optimaalisen esitystavan tunnistaminen. Tutkimuksen sovellusosassa otettiin kantaa ainoastaan yhteen kytkimen toimintaan liittyvän parametrin tallentamiseen. Järjestelmässä on paljon muitakin kytkimiä sekä muita osia, joiden toiminnan dokumentointia kannattaisi tutkia. Jatkotutkimuksessa ja esitetyn ratkaisun tuotantokäyttöversion kehittämisessä tulee tarkentaa, mihin käyttötarkoitukseen dataa kerätään. Toteutetussa sovelluksen versiossa kaikki sovelluksen osat toimivat samalla palvelimella, mikä ei olisi tarkoituksenmukaista tuotantokäyttöön otettavassa versiossa. Ratkaisun kehittämisessä haastavaa oli valita sopiva laajuus, jonka toteuttaminen ja dokumentointi välittäisivät lukijalle aihepiirin oleellisimmat haasteet hyvin ilman, että aihealue kasvaa liian suureksi.

4 Tutkimuksen tulokset

Tutkittaessa muihin aihepiireihin kuin Sovelia Core PLM:ään löydettiin menetelmiä, jotka ovat sovellettavissa tutkimuksen pääaihealueena olevaan Sovelia Core PLM -järjestelmään. Tiedonlouhinta osoittautui prosessiksi, jonka menetelmiä voidaan soveltaa PLM-järjestelmän automaattisen dokumentoinnin ratkaisun kehittämisessä. Tutkimuksen kirjallisuuskatsausvaiheessa löydettyjä menetelmiä soveltamalla kehitettiin pieni rajattu ratkaisu, joka osoittaa, että Sovelia Core PLM:stä voidaan nykyisellä konfiguraatiollakin tallentaa sellaista tietoa, jota ei ole tähän mennessä tallennettu ja jonka tallentamisesta olisi hyötyä asiakaskohtaisen järjestelmäasennuksen kehitystyön tueksi. Kytkimistä voitaisiin saada talteen paljon muutakin mielenkiintoista tietoa, jos niiden suunnittelussa otettaisiin erityisesti huomioon, millaisia tarpeita niiden toiminnan seuraamiseen liittyy.

4.1 Testien tulokset ja havainnot

Kehitetty automaattinen dokumentointiratkaisu toteuttaa tehokkaasti kytkimen päiväkohtaisten ajojen laskennan, tallennuksen sekä esittämisen visuaalisessa muodossa. Valitut tekniikat ratkaisun toteuttamiseen osoittautuivat toimiviksi. Pythonilla toteutettu ohjelma, joka laskee määritellyn parametrin sisältävien rivien määrät lokitiedostosta ja tallentaa tiedon yhdessä lokitiedoston päiväyksen kanssa, toimii tarkoituksessaan hyvin. Chart.js-visualisointikirjastoa hyödyntävä kerätyn datan visualisointiohjelma hakee esitettävät tiedot JSON-tiedostosta nopeasti ja esittää tiedon selkeästi verkkoselaimessa. Täysin automatisoituna parametrin lokitiedostossa esiintymisen automaattinen dokumentointi säästää aikaa sitä enemmän, mitä pidemmällä ajalla sitä tehdään, mitä enemmän parametreja seurataan ja mitä useammassa järjestelmässä dokumentointia tehdään.

4.2 Parannusehdotukset ja jatkokehitys

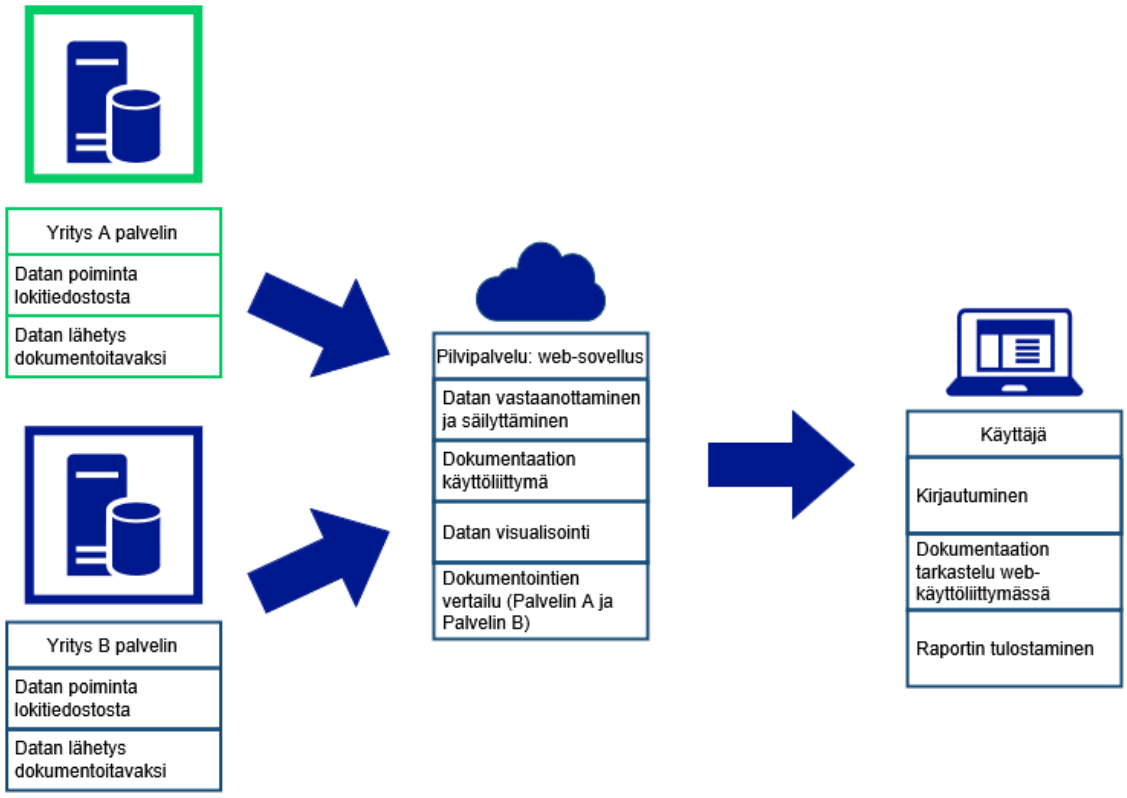
Toteutettu ratkaisu kerää hyvin yksinkertaista tietoa: kytkimen ajomääriä vuorokaudessa. Jos kytkimen toiminnan seuraamiseen liittyvät vaatimukset olisi määritelty tarkemmin, pystyttäisiin ratkaisu suunnittelemaan niiden tarkkojen vaatimusten mukaisiksi. Opinnäytetyössä toteutettua ratkaisua ei ole automatisoitu täysin. Se täytyy ajaa käsin esimerkiksi komentoriviltä. Ratkaisun lokitiedosta kerättävän tiedon keräysajon automatisointi esimerkiksi Task Scheduler-ohjelmalla veisi ratkaisun automaation tasoa pidemmälle. Toteutettu ratkaisu toimii esimerkkinä, mutta sen käyttökohteet sellaisenaan hyödynnettäväksi ovat rajalliset. Jatkokehityksessä kytkimen toiminnan mittareiden lisäksi tiedon tallennus- sekä esitystapoja voitaisiin kehittää. Myös ratkaisun saavutettavuudessa eri käyttäjille olisi parannettavaa. Paikallisesti palvelimelle tallennettava JSON-tiedosto ei suuressa mittakaavassa palvele käyttötarkoitusta täysmittaisesti, vaan JSON-tiedostoon tallennettu tieto pitäisi koota Sovelia Core PLM-

palvelimilta keskitetysti yhteen paikkaan, kuten pilvipalveluun. Kun JSON-tiedostot kerättäisiin yhteen paikkaan, eri järjestelmistä kerättyjen tietojen vertaileminen keskenään olisi tehokasta. Kuvassa 16 esitellään toteutettua ratkaisua pidemmälle viety automaattisen dokumentoinnin arkkitehtuuri, joka soveltuu tuotantokäyttöön. Arkkitehtuurin osia ei kannattaisi rakentaa alusta lähtien itse, vaan sen toteuttamisessa kannattaisi hyödyntää olemassa olevia web-teknologioita. Esimerkiksi Azure-pilviratkaisut tarjoaisivat hyviä valmiita komponentteja tiedon siirtämiseen palvelinten välillä sekä työkaluja web-sovelluksen isännöintiin. Azuren tarjoamat ominaisuudet saattaisivat riittää koko ratkaisun tuotantoversion tekemiseen, jos tässä opinnäytetyössä esitetyt vaatimukset ovat sille riittävät. Tarvittaessa, esimerkiksi ulkopuolisille käyttäjille tietojen saataville saattamiseksi käyttöliittymä voitaisiin toteuttaa myös Azuresta erillisenä web-sovelluksena. Web-sovellus voitaisiin toteuttaa jollain suositulla web framework stackilla, kuten Node.js ja Express.js yhdistelmällä tai Djangolla.

Kuvan 16 ratkaisussa aiemmassa kuvan 9 esimerkissä esiteltyjen yritysten A ja B palvelimilla kerättäisiin data lokitiedostoista ja lähetettäisiin se pilvipalveluun. Pilvipalvelussa isännöitäisiin web-sovellusta, jonka backendissa säilytettäisiin dokumentointidataa. Frontendin toteuttamiseen olisi kaksi vaihtoehtoa:

1. toteutus Azure Portalissa sen tarjoamilla komponenteilla ja työkaluilla
2. web-sovellus toteutettuna jollain muulla web frameworkilla

Kuva 16. Datan käsittelyn arkkitehtuuri



Azure-sovellus voisi käyttää Monitor-ominaisuutta, jolla saadaan luotua helposti kuvaajia. Tämä vaihtoehto olisi myös tietoturvan kannalta hyvä, sillä Azuren tarjoavan Microsoftin kaksivaiheinen kirjautuminen olisi automaattisesti käytössä. Azure Monitor saattaa kuitenkin olla paremmin monitorointiin kuin dokumentointiin soveltuva työkalu. Vaihtoehtoinen toteutustapa olisi web-sovellus, johon käyttäjä kirjautuu erillisillä tunnuksilla. Kirjautumisen jälkeen hän voi valita dokumentaation, jota haluaa tarkastella. Molemmissa vaihtoehtoisissa toteutustavoissa käyttäjän tulisi voida valita tarkasteltava dokumentaatio, josta näytettäisiin visualisointi. Erilaisia visualisointeja voisi olla valittavissa useita erilaisia esimerkissä käytetyn pylväsdiagrammin lisäksi. Käyttäjä voisi myös vertailla eri ajanjaksoilta sekä eri järjestelmistä tallennettuja dokumentointeja keskenään, sekä tulostaa niistä raportteja.

5 Johtopäätökset

Tämän opinnäytetyön tavoitteena oli kehittää automaattinen dokumentointijärjestelmä Sovelia Core PLM -asennusten muutosten dokumentointiin. Tutkimus osoitti, että tällaisen järjestelmän toteuttaminen on mahdollista ja hyödyllistä palveluorganisaation toiminnan tehostamiseksi. Toteutetussa automaattisessa dokumentoinnissa esimerkkinä käytetty datatyyppi on sellainen, että sen keräämisen automatisointi näyttäytyy hyvin kannattavana, jos tietoa halutaan ylipäätään kerätä. Sitä ei kannata tehdä käsin, sillä tiedon keräämisen automatisointi on suhteellisen yksinkertaista, ja pidemmän päälle automatisoinnin rakentamiseen käytetty aika säästää aikaa verrattuna tiedon käsin keräämiseen. Mitä pidemmältä ajalta ja mitä useampia parametreja koskevaa dataa kerätään, sitä enemmän aikaa suhteessa säästetään.

Opinnäytetyössä käytettiin lähteinä erityisesti yritysten tuottamia tekstejä tieteellisten tutkimusten sijaan siksi, että lähteiden etsinnässä yritysten julkaisemat artikkelit nousivat tieteellisiä artikkeleja voimakkaammin esiin ja tarjosivat paremmin vastauksia opinnäytetyön kannalta oleellisiin kysymyksiin. Tämä voi johtua siitä, että alan yritykset ratkovat konkreettisia asiakkaiden sekä omiin prosesseihin liittyviä ongelmia. Sen sijaan tieteellisissä artikkeleissa tutkimuskohteena on usein abstraktimpi aihe. Siten tämän työn palvellessa PLM-järjestelmää toimittavaa yritystä on loogista, että sovellettavia menetelmiä ongelmanratkaisuun löytyy toisten tuottavan teollisuuden yritysten kanssa toimivien yritysten julkaisemista artikkeleista. Kirjat *Data Science for Business* sekä *The Data Warehouse Toolkit* ovat poikkeuksia opinnäytetyön lähteissä, ja ne tarjoavat hyödyllistä tietoa jatkotutkimuksiin opinnäytetyön aihepiirissä.

M. Bossonin artikkelissa tuotantodatan keräämisestä keskityttiin tuottavan teollisuusyrityksen fyysisestä valmistusprosessista kerättävään dataan. A. Twinin artikkelissa tiedonlouhinnasta tuotantodata oli yksi esiteltyjen menetelmien sovelluskohteista. Tämän opinnäytetyön pääaihealue eroaa näistä esimerkeistä

siten, että kyse on tuottavan teollisuuden prosesseissaan hyödyntämän järjestelmän sisäisen tiedon keräämiseen liittyvä, mutta ei välttämättä suoraan tuotantodataan liittyvä tutkimus. Tässä opinnäytetyössä esitettyä ratkaisua voidaan soveltaa myös tuotannon prosessien kehittämiseen, mutta se ei ole sen ensisijainen tarkoitus. Tutkimus toteutettiin ensisijaisesti PLM-järjestelmän kehittämiseksi, joka voi välillisesti auttaa myös tuottavan teollisuuden tuotantoprosessien kehittämisessä.

Tutkimuksen tulokset vahvistavat, että Sovelia Core PLM -järjestelmästä voidaan kerätä arvokasta tietoa asiakaskohtaisten asennusten kehittämiseksi nykyisellä konfiguraatiolla. Toteutettu Python-pohjainen lokianalysaattori ja JavaScript-visualisointityökalu osoittautuivat tehokkaiksi ratkaisuiksi kytkimien päiväkohtaisten ajojen laskemisessa, tallentamisessa sekä visualisoinnissa.

Automaattinen seurantajärjestelmä tarjoaa merkittäviä etuja verrattuna aiempiin manuaalisiin dokumentointimenetelmiin:

1. Parempi tarkkuus ja kattavuus muutosten seurannassa
2. Ajan säästö rutiinitehtävissä
3. Helpompi pääsy vanhoihin tietoihin
4. Mahdollistaa trendien havaitsemista sekä tulevien tarpeiden ennakointia

Tutkimus paljasti myös kehityskohteita ja jatkotutkimusaiheita:

1. Lokitiedostoihin kirjoitettava tieto voitaisiin optimoida tukemaan paremmin automaattista dokumentointia
2. Tiedon tallennusmenetelmiä voitaisiin kehittää skaalautuvuuden ja tehokkuuden parantamiseksi
3. Ratkaisun automaatiotasoa voitaisiin nostaa entisestään
4. Käyttöön otettava versio tulisi toteuttaa useita palvelimia hyödyntäen (kuva 16), poiketen toteutetusta esimerkkiratkaisusta

Yhteenvetona voidaan todeta, että automaattinen dokumentointijärjestelmä tarjoaa lisäarvoa Sovelia Core PLM -asennusten hallintaan ja kehittämiseen. Järjestelmä mahdollistaa tehokkaamman resurssien

käytön ja paremman ymmärryksen asiakaskohtaisista tarpeista, mikä puolestaan mahdollistaa asiakastyytyväisyyden parantamista.

Tutkimuksessa kehitetty automaattinen dokumentointijärjestelmä vastaa päätutkimuskysymykseen tarjoamalla tehokkaan tavan kerätä muutoksiin liittyvää dataa, mikä tehostaa asiakaskohtaisten järjestelmäasennusten kehitystä asiakkaiden muuttuvien tarpeiden täyttämiseksi.

6 Pohdinta

Sovelia Core PLM-järjestelmän tuottamaa datamassaa voi ja kannattaa hyödyntää asiakaskohtaisten järjestelmämuutosten dokumentointiin. Hyvin suunniteltu dokumentointi tarjoaa paljon mahdollisuuksia järjestelmäasennuksen kehittämiseen, mikä taas on hyödyllistä asiakkaan muuttuviin tarpeisiin vastaamiseksi. Muista PLM-järjestelmien ulkopuolisten tiedon keräämiseen kehitettyjen menetelmien joukosta löytyi lukuisia automaattisessa dokumentoinnissa potentiaalisesti hyödynnettäviä menetelmiä, joista vain pientä osaa hyödynnettiin ratkaisussa. Vaatimusmäärittelyn tarkentuessa ratkaisua voidaan kehittää eteenpäin tarvittavaan suuntaan. Jatkotutkimuksessa dokumentoinnin tarpeiden tarkentuessa voitaisiin syventyä nyt esitettyä edistyneempiä menetelmiä käyttävän ratkaisun kehittämiseen.

Tämä opinnäytetyö tarjoaa hyvän pohjan automaattisten dokumentointimenetelmien jatkokehitystä varten. Jatkotutkimuksen kannalta on useita kiinnostavia teemoja, joihin liittyviä menetelmiä kannattaisi tutkia. Näitä teemoja ovat data engineering-viitekehykseen kuuluvat pilvipohjaiset datajärvet ja datavarastot sekä edistyneet dataputkiarkkitehtuurit, koneoppimisen viitekehyksessä poikkeamien havaitseminen järjestelmälokeista sekä aikasarjatietokannat järjestelmämuutosten seurantaan. Opinnäytetyössä yhdistettiin opinnoissa sekä työelämässä Sovelia Core PLM:n parissa opittuja taitoja. Sen toteutusvaiheessa syvennettiin osaamista PLM-järjestelmien parissa. Opinnäytetyössä havainnoitiin PLM-järjestelmää uusilla tavoilla, sillä siinä tekijän täytyi ensimmäistä kertaa suunnitella omaa uutta toiminnallisuutta vanhojen olemassa olevien käyttämisen ja konfiguroinnin sijaan. Opinnäytetyön prosessi avasi uusia ajattelutapoja ongelmanratkaisuun työssäni. Kehityin tekniikan ammattilaisena ja kirjoittajana. Työssä onnistuttiin vastaamaan tavoitteenasettelussa annettuihin kysymyksiin, tosin kerätyn datan potentiaalisiin käyttötarkoituksiin vain suppeasti. Vähäiset lähtötiedot potentiaalisista käyttötarkoituksista huomioiden kysymyksiin vastaamisessa onnistuttiin hyvin. Tämä opinnäytetyö toimii hyvänä pohjana jatkotoimille, joissa potentiaalisten käyttötarkoitusten kysymykseen etsitään tarkempaa vastausta. Sitten kun selkeä vastaus tuohon kysymykseen löytyy, työ varsinaisen käyttöön otettavan automaattisen dokumentointityökalun kehittämiseksi voidaan aloittaa täysimittaisesti.

Lähteet

Airbyte. (6.12.2024). 10 Interesting Data Collection Techniques & Methods For 2025. https://airbyte.com/data-engineering-resources/data-collection-techniques

Aras. (2022). *Case study – Clean Air, Clear Data* https://www.fulvisol.com/wp-content/uploads/2022/08/2736-CS-tamturbo-2.pdf

Bosson, M. (19.9.2023). *Manufacturing Data Collection: How to Get Starterd in 5 Easy Steps.* Factbird. https://www.factbird.com/blog/manufacturing-data-collection

Hashemi-Pour, C. (n.d.). *Big data.* Techtarget. https://www.techtarget.com/searchdatamanagement/definition/big-data

Kenton, W. (27.4.2024). *Manufacturing: Definition, Types, Examples, and Use as Indicator*. Investopedia. https://www.investopedia.com/terms/m/manufacturing.asp

Kimball, R. Ross, M. (2013). *The Warehouse Toolkit The Definitive Guide to Dimensional Modeling Third Edition.* Wiley. (443-496)

Morrison, C. (1.1.2025). *Manufacturing ERP: The Top 10 ERP Systems for 2025*. https://www.top10erp.org/blog/manufacturing-erp

Provost, F. Fawcett, T. (2013). *Data Science for Business What You Need to Know About Data Mining and Data-Analytic Thinking.* O'Reilly. (19-34)

Rautelin, I. (25.2.2025). *Enigneering change request (ECR) – Overview.* Symetri https://help.sovelia.com/docs/overview-1

Rautelin, I. (24.10.2022). *Engineering change orders (ECO) – Overview.* Symetri https://help.sovelia.com/docs/overview-eco

Ray, T. (26.2.2025). *Items, drawings and product documents – Lifecycle.* Symetri https://help.sovelia.com/docs/lifecycle-states

Roima. (n.d.). *Why Product Lifecycle Management and Enterpsise Resource Planning Belong Together.* https://www.top10erp.org/blog/manufacturing-erp

Symetri. (n.d.). *Europress manufactures modern waste compactors and balers using the Sovelia Core PLM system and Sovelia Configurator.* https://sovelia.com/customers/sovelia-core-europress/

Tuychiev, B. (2.2.2025). Automated Data Collection – A Comprehensive Guide. Firecrawl. https://www.firecrawl.dev/blog/automated-data-collection-guide

Twin, A. (23.2.2024). *What is Data Mining? How It Works, Benefits, Techniques, and Examples.* Investopedia. https://www.investopedia.com/terms/d/datamining.asp

3R. (n.d.). *Enchancing Data Flow and Collaboration through PLM Integration with CAD and ERP.* https://3rinfo.com/enhancing-data-flow-and-collaboration-through-plm-integration-with-cad-and-erp/

Liite 1. Patterncounting.py

```
import re
import json
import sys

#run from cmd with python patterncount-definitive.py /path/to/your/logfile.log
def get_first_date(file_path):
    """Extract the first date from a log file in YYYY-MM-DD format."""
    datetime_pattern = re.compile(r"(\d{4}-\d{2}-\d{2}) \d{2}:\d{2}:\d{2}")

    with open(file_path, 'r', encoding='utf-8') as file:
        for line in file:
            datetime_match = datetime_pattern.search(line)
            if datetime_match:
                return datetime_match.group(1)  # Return only the date part

    print("Warning: No valid date found in the file.")
    return None

def count_pattern_occurrences_in_file(file_path, pattern):
    """Count the number of lines containing the specified pattern."""
    compiled_pattern = re.compile(pattern)
    unique_line_count = 0

    with open(file_path, 'r', encoding='utf-8') as file:
        for line in file:
            if compiled_pattern.search(line):
                unique_line_count += 1

    return unique_line_count

def load_data_from_json(json_file_path):
    """Load existing data from a JSON file."""
    try:
        with open(json_file_path, 'r', encoding='utf-8') as json_file:
            data = json.load(json_file)
            if isinstance(data, list):
                return data
            else:
                print("Warning: JSON file is not a list. Starting with an empty list.")
                return []
    except FileNotFoundError:
        return []

def save_data_to_json(data_list, json_file_path):
    """Save data to a JSON file."""
    with open(json_file_path, 'w', encoding='utf-8') as json_file:
```

```
        json.dump(data_list, json_file, indent=4)

# Default configuration
pattern = r'plm_check_attributes'
json_file_path = 'pattern_data.json'

# Get file path from command line argument, or use default if not provided
if len(sys.argv) > 1:
    file_path = sys.argv[1]
else:
    file_path = 'tomcat9-stdout.xxxx-xx-xx.log' # replace with the actual log filepath
    print(f"No file path provided. Using default: {file_path}")

# Get the first date occurrence in the file
first_date = get_first_date(file_path)

if first_date is not None:
    # Count unique line occurrences of the pattern
    count = count_pattern_occurrences_in_file(file_path, pattern)

    # Load existing data from JSON
    existing_data = load_data_from_json(json_file_path)

    # Prepare new data entry
    new_entry = {
        "date": first_date,
        "pattern": pattern,
        "count": count
    }

    # Append new entry to the existing data
    existing_data.append(new_entry)

    # Save updated data back to JSON
    save_data_to_json(existing_data, json_file_path)

    print(f"Pattern '{pattern}' occurred in {count} unique lines starting from first date
'{first_date}', and details were appended to {json_file_path}.")
else:
    print("No valid date found; data was not processed.")
```

Liite 2. index.html

```
<!DOCTYPE html>
<html lang="en">
<head>
    <meta charset="UTF-8">
    <meta name="viewport" content="width=device-width, initial-scale=1.0">
    <title>JSON Data Visualization</title>
    <script
src="https://cdnjs.cloudflare.com/ajax/libs/Chart.js/4.0.1/chart.umd.min.js"></script>
</head>
<body>
    <div style="width: 60%; margin: auto;">
        <canvas id="barChart"></canvas>
    </div>

    <script>
        async function fetchDataAndRenderChart() {
            try {
                const response = await fetch('pattern_data.json'); // Fetch the JSON data from
the file
                const jsonData = await response.json();

                // Get date labels and data counts
                const labels = jsonData.map(entry => entry.date);
                const dataCounts = jsonData.map(entry => entry.count);

                // Create the bar chart
                const ctx = document.getElementById('barChart').getContext('2d');
                new Chart(ctx, {
                    type: 'bar',
                    data: {
                        labels: labels,
                        datasets: [{
                            label: 'Count of plm_check_attributes',
                            data: dataCounts,
                            backgroundColor: 'rgba(55, 166, 240, 0.5)',
                            borderColor: 'rgba(55, 166, 240, 1)',
                            borderWidth: 1
                        }]
                    },
                    options: {
                        scales: {
                            y: {
                                beginAtZero: true
                            }
                        }
                    }
```

```
            });
        } catch (error) {
            console.error("Error fetching or parsing JSON data:", error);
        }
    }

    // Load the data and render the chart when the page loads
    window.onload = fetchDataAndRenderChart;
  </script>
</body>
</html>
```

Liite 3. Pattern_data.json


```
[
  {
    "date": "2025-02-17",
    "pattern": "plm_check_attributes",
    "count": 0
  },
  {
    "date": "2025-02-18",
    "pattern": "plm_check_attributes",
    "count": 0
  },
  {
    "date": "2025-02-19",
    "pattern": "plm_check_attributes",
    "count": 1
  },
  {
    "date": "2025-02-20",
    "pattern": "plm_check_attributes",
    "count": 19
  },
  {
    "date": "2025-02-21",
    "pattern": "plm_check_attributes",
    "count": 3
  },
  {
    "date": "2025-02-22",
    "pattern": "plm_check_attributes",
    "count": 0
  },
  {
    "date": "2025-02-22",
    "pattern": "plm_check_attributes",
    "count": 0
  }
]
```